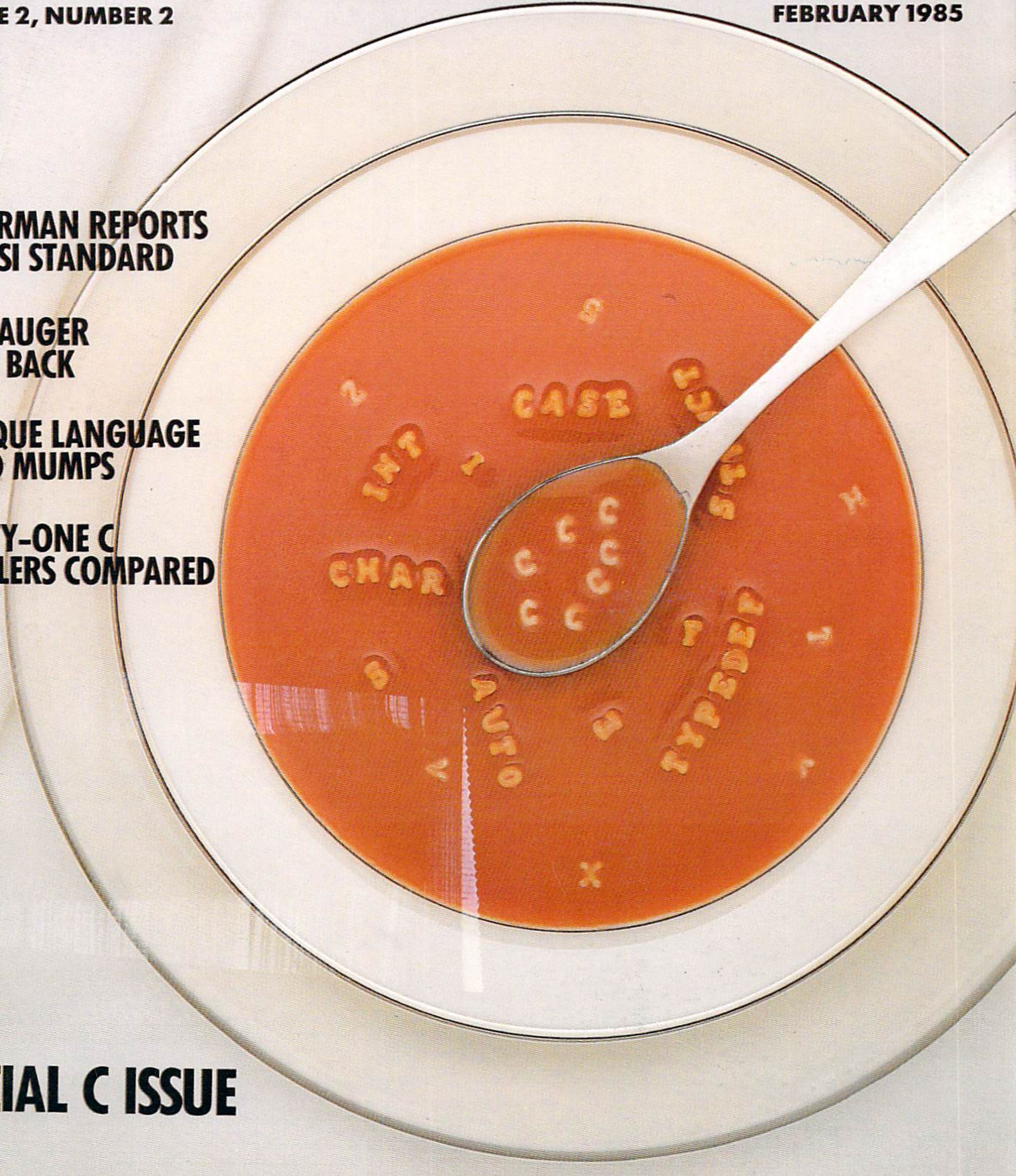
COMPUTER LANGUAGE™

$2.95

VOLUME 2, NUMBER 2

FEBRUARY 1985


CASE
INT
CHAR
AUTO
TYPEDEF

C CHAIRMAN REPORTS ON ANSI STANDARD

P.J. PLAUGER LOOKS BACK

A UNIQUE LANGUAGE CALLED MUMPS

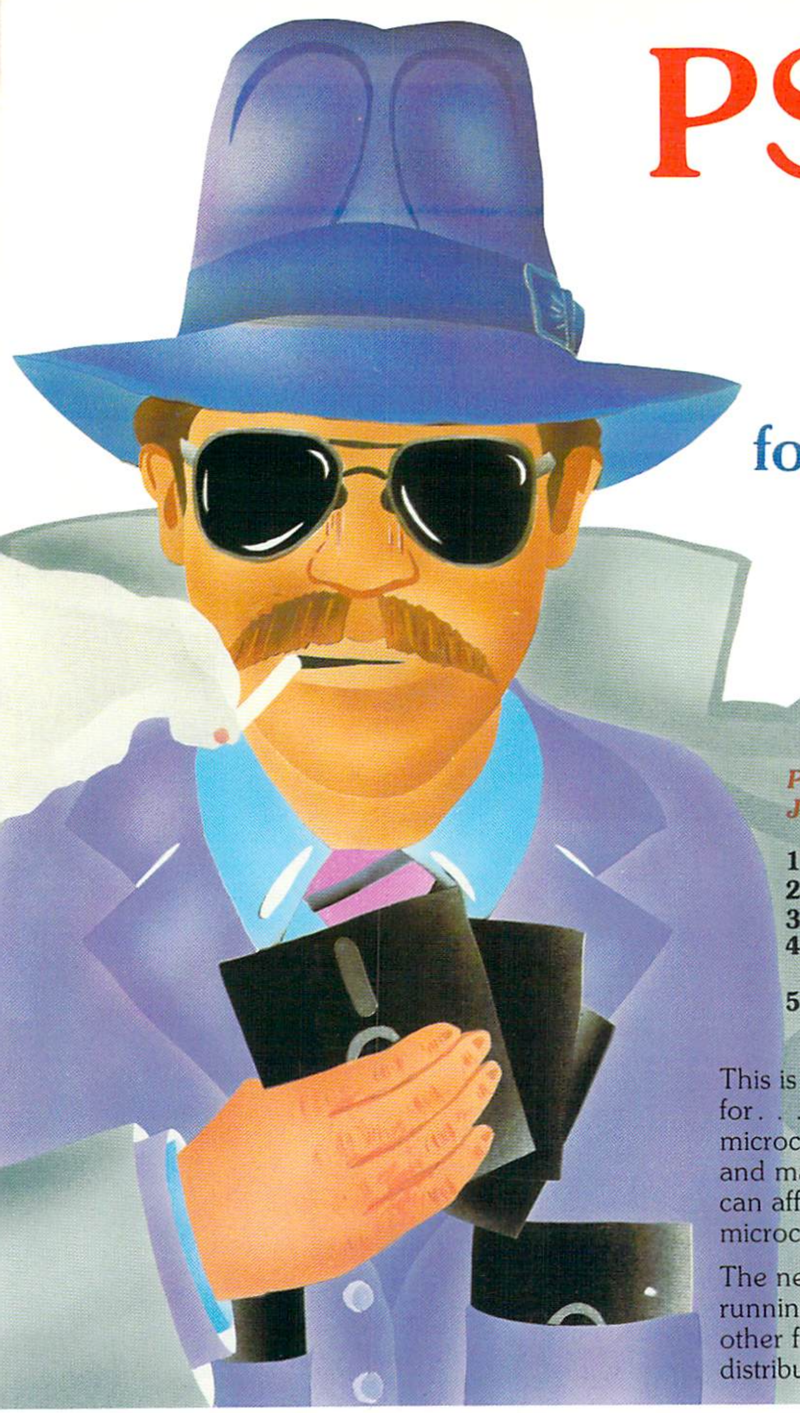
TWENTY-ONE C COMPILERS COMPARED

SPECIAL C ISSUE

PSSST...

JANUS/Ada

for $99.95!!



PRESENTING THE NEW JANUS/Ada C-PAK!!

1. **Janus/Ada Compiler**
2. **Janus/Ada Linker**
3. **Janus/Ada Libraries**
4. **Janus/Ada Example/ Programs**
5. **Janus/Ada User Manual**

AND THESE ADDED FEATURES!!

1. **Free User's Group**
2. ***$99.95 Discount* on the Janus/Ada D-Pak**
3. **No License!!**
4. **No Copy Protection!!!**
5. **Customer tested for over 3 years!!!**

This is the introductory Ada™ package you've been waiting for . . . over three years of actual field use, specifically on microcomputers, by the government, Fortune 500 businesses and major universities. Realistically priced, at $99.95, so *you* can afford the most popular Ada implementation used on microcomputers!

The new "C"-Pak is available for most microcomputers running MS-DOS, including the IBM PC AT™, as are all the other fine Janus/Ada programs. Call us or an authorized distributor for your copy today!

National Distributors

Westico, Inc.
25 Van Zant St.
Norwalk, CT 06855
(203) 853-6880

ASH II
7407 Marisol
Houston, TX 77083
(713) 933-1828

A.O.K. Computers
816 Easley St., Suite 615
Silver Springs, MD 20910
(301) 588-8446

Trinity Solutions
5340 Thornwood Dr., Suite 102
San Jose, CA 95123
(408) 226-0170

MicroProgramming, Inc.
P.O. Box 3356
Chatsworth, CA 91313
(818) 993-6475

International Distributors

Ada Australia
218 Lutwyche Rd.
Windsor 4030
QLD. Australia
(07) 57 9997

Progesco
155, rue du Faubourg
St. -Denis
75010 Paris
France
(1) 205.39. 47

Lifeboat, Inc. Japan
3-6, Kando-Nishikicho
Chiyoda-ku
Tokyo 101, JAPAN
03-293-4711

CP/M, CP/M-86, CCP/M-86 are trademarks of Digital Research, Inc.
*ADA is a trademark of the U.S. Department of Defense
MS-DOS is a trademark of Microsoft

©Copyright 1984 RR Software

RR SOFTWARE, INC.

specialists in state of the art programming

P.O. Box 1512 Madison, Wisconsin 53701
(608) 244-6436 TELEX 4998168

CIRCLE 58 ON READER SERVICE CARD

IDRIS
PASCAL
C COMPIL
A LA CARTE

Make Whitesmiths, Ltd. Part of Your 1985 Software Strategy.

For six years, software developers and systems integrators have looked to Whitesmiths, Ltd. for technologically superior compilers and multi-tasking operating systems.

Before you make your next move, contact Whitesmiths, Ltd.

Whitesmiths, Ltd.
97 Lowell Road
Concord, MA 01742
TLX 951708 SOFTWARE CNCM.

Whitesmiths, Ltd.

DISTRIBUTORS: Australia, Fawnray Pty. Ltd., Hurstville, (612) 570-6100; **Japan** Advanced Data Controls Corp., Chiyoda-ku, Tokyo (03) 263-0383; **United Kingdom,** Real Time Systems, Douglas, Isle of Man 0624-26021; **Sweden,** Unisoft A.B., Goteborg, 31-125810.

CIRCLE 85 ON READER SERVICE CARD

SQUEEZE MORE OUT OF EVERY ON-LINE MINUTE.

CompuServe Software
VIDTEX
CompuServe

WITH NEW VIDTEX™ COMMUNICATIONS SOFTWARE FROM COMPUSERVE.

Presenting the software package that makes your computer more productive and cost-efficient.

CompuServe's new Vidtex™ is compatible with many personal computers sold today (including Apple,® Commodore,® and Tandy/Radio Shack® brands). And it offers the following features*–and more–to let you communicate more economically with most time-sharing services (including CompuServe's Information Service).

Auto-Logon. Lets you log on to a host simply and quickly by utilizing prompts and responses defined by you. Also allows quick transmission of predefined responses to host application programs after logging on.

Function Keys. Let you consolidate long commands into single keystrokes. Definitions can be saved to and loaded from disk file, allowing multiple definitions for multiple applications.

Error-Free Uploading and Downloading. CompuServe "B" Protocol contained in Vidtex lets you transfer from your computer to CompuServe and from CompuServe to your computer anywhere in the country. Also provides error-free downloading from CompuServe's extensive software libraries.

Full Printer Support. Printer buffer automatically buffers characters until printer can process; automatically stops on-line transmission when full; and automatically resumes transmission when capacity is re-established. Also, lets you print contents of textual video screen or RAM buffer at any time.

Capture Buffer. Saves selected parts of a session. Contents can be written to a disk file; displayed both on and off line; loaded from disk; and transmitted to the host.

On-line Graphics. Integral graphics protocol displays stock charts, weather maps and more.

If you are already a CompuServe subscriber, you can order Vidtex on line by using the GO ORDER command. Otherwise, check with your nearest computer dealer; or to order direct, call or write:

CompuServe

P.O. Box 20212, 5000 Arlington Centre Blvd.
Columbus, Ohio 43220

1-800-848-8199
In Ohio, call 614-457-0802

An H&R Block Company

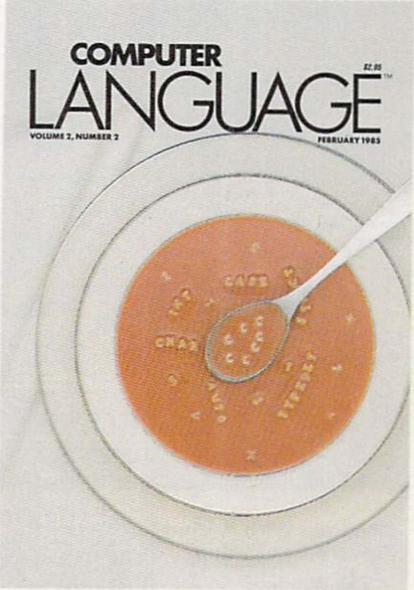
CIRCLE 7 ON READER SERVICE CARD

---

*Some versions of the Vidtex software do not implement all features listed.

Vidtex is a trademark of CompuServe, Incorporated. Apple is a trademark of Apple Computer, Inc. Commodore is a trademark of Commodore Business Machines. Radio Shack is a trademark of Tandy Corp.

COMPUTER LANGUAGE



ARTICLES

The Standardization of C 26
by Jim Brodie
As a language well suited for both applications and systems programming, C's popularity has prompted the development of a standard to further enhance the language's portable syntax. Jim Brodie, chairman of ANSI's C Programming Language Standards Committee, discusses the history behind the standardization effort, the goals and working principles of the committee, and how the language currently being defined is different from the de facto, Kernighan and Ritchie standard.

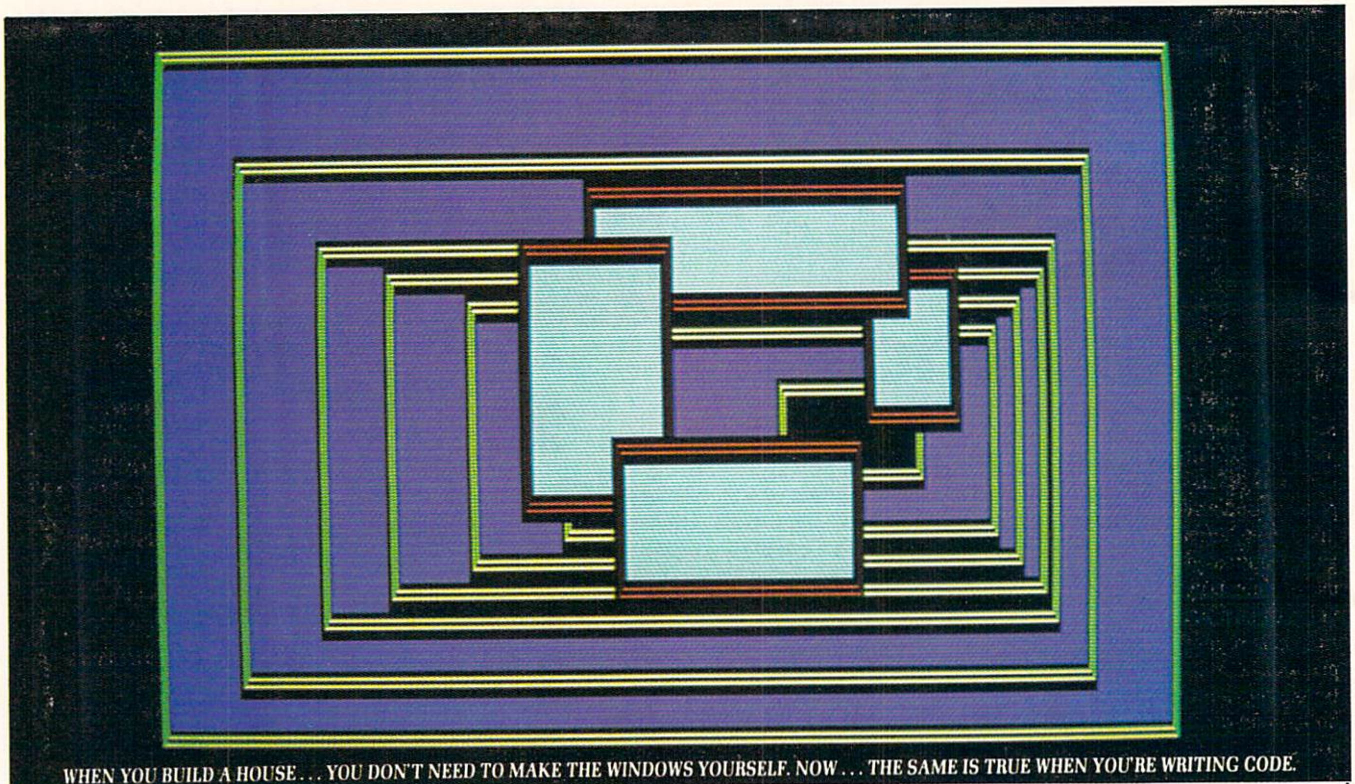
C Instead of FORTRAN? 33
by Anthony Skjellum
Because of FORTRAN's installed base of users and its limited ability to be portable between hardware configurations, it has traditionally been the default language for scientific and engineering programming tasks. However, despite recent movements in the industry to give FORTRAN a face-lift by adding certain structured language constructs, C may better answer the need for an efficient, modular, and structured programming environment for scientific and engineering projects.

Programming Macros in C, Part I 43
by Alexander B. Abacus
In this three-part series, Alexander Abacus presents various methods and techniques for using macros to improve C program development efficiency. Part I focuses on how to use macros as building blocks for parsing command line arguments. In the following two installments, we'll see how macros can also be used in debugging large programs and as a means for translating the common representation of data into different but related tables required by different programs.

C to Assembly Interface 49
by Brian H. Burger
When programming large, complex projects, it often becomes necessary to access architecture-dependent machine capabilities to increase program speed and efficiency. Yet to some programmers, the C language can open a Pandora's box since it requires more knowledge of the machine than most other languages.

DEPARTMENTS

Publisher's Notes 5
Feedback 9
CrossXthoughts 13
A new column on programming methods and techniques
Designers Debate 17
UNIX on micros
ComputerVisions 21
P.J. Plauger reflects on the history of C
Exotic Language of the Month Club 61
MUMPS: A multi-user, data management language
Product Bingo 71
New product announcement column
Software Review 73
Special *COMPUTER LANGUAGE* C compiler analysis
Advertiser Index 104



WHEN YOU BUILD A HOUSE... YOU DON'T NEED TO MAKE THE WINDOWS YOURSELF. NOW... THE SAME IS TRUE WHEN YOU'RE WRITING CODE.

Windows With A View Toward The Future

The Window Machine™ occupies only 12K! Written in tight, fast Assembler, it performs like a racing engine... with more power than you'll probably ever need. Yet, it's an engine designed to fit in the vehicle of your choice... from a "stripped-down" 128K IBM PC to a fully loaded AT. The programs you write today will run on the broadest range of machines possible... now, and in the future.

Windows Bigger Than Your Screen?

Here's where the VSI part of our name fits in. VSI means Virtual Screen Interface. Behind each window, there's a much bigger picture. VSI defines virtual screens rather than just windows. The window itself shows whatever portion of its virtual screen you wish to exhibit at any given point in your program. Each screen can be up to 128 x 255 (columns x rows, or rows x columns). And there are more than 100 screen primitives at your command.

Multilingual Windows

You can order The Window Machine with the language interface of your choice: C, Pascal, Compiled Basic, Fortran, Cobol, or PL1. We've even recently completed an interface for Turbo Pascal*, so that now true, full-featured windowing can be utilized with this fine compiler. (Turbo's own built-in "windowing" procedure is extremely limited).

> *These are coders' windows... designed to be built into the programs you are writing. They can overlap, move anywhere on the screen, grow, shrink, vanish or blink. They can be bordered in anything from a simple line to flashing asterisks... or even no border at all. And you can have up to 255 of them at a time! Color or monochrome ...of course!*

Why did Simon & Schuster, 3Com, Tymshare, and Revlon choose VSI – The Window Machine?

(and how come you can buy it for such a low price?)

$59.95

Windows That Won't Break You

We decided to save you a lot of money. So, we left behind fancy binders, monogrammed slip cases and plastic presentation boxes. Instead, you'll find an extremely powerful tool and a 200 page manual written with an eye toward simplicity, clarity and completeness. (We figured if you wanted ribbons and bows you could always add them yourself.)

And by offering you the product ourselves, we were able to cut out all the middlemen and save you a tremendous amount of money.

*Turbo Pascal is a Trademark of Borland International

VSI *The Window Machine*™

Available for the IBM PC, XT, AT, IBM Compatibles, and the Wang, T.I., HP 150.

The Window Machine Includes:

- *Zoom Windows*
- *Multiple Virtual Screens (up to 255)*
- *Choice of Borders (including flashing borders)*
- *Support for all Color and Monochrome Video Attributes (no graphics card required)*
- *Built-in Diagnostics*
- *And much, much more*

ORDER YOUR COPY OF VSI—THE WINDOW MACHINE TODAY For Visa and MasterCard orders call toll free:

1-800 742-2500

The Window Machine™ $59.95 Shipping and handling included

LANGUAGE INTERFACE:

☐ Lattice C ☐ Realia Cobol ☐ Microsoft Basic Compiler ☐ Microsoft Fortran
☐ PL1 ☐ Microsoft Pascal ☐ Turbo Pascal (full featured true windowing)

COMPUTER ____________

Name ____________

Address ____________

City ____________ State ____ Zip Code ________

☐ Check ☐ Money Order ☐ VISA ☐ MasterCard

Card # ____________ Exp. Date ________

*California residents: tax included. Orders outside the USA: please add $5 for shipping and handling.

CL

AMBER SYSTEMS
1171 S. Saratoga-Sunnyvale Road
San Jose, CA 95129

AMBER SYSTEMS, INC. 1171 S. Saratoga-Sunnyvale Road, San Jose CA 95129

FOR DEALER INQUIRIES: CALL OUR 800 NUMBER

CIRCLE 2 ON READER SERVICE CARD

Publisher's Notes

Welcome to our first theme issue! This month we spotlight the C programming language.

The amount of reader interest and increased popularity of C made the subject choice of this issue an easy one. The C language has endured the evolutionary process that any new language must undergo when its followers attempt to gather industry-wide acceptance for the use of their language. After the past 15 years or so since its birth at Bell Laboratories, C is regarded today as perhaps the premier language for systems and applications programming because of certain built-in features such as the portability of source code, its simple but structured syntax, and its machine-level interface capabilities.

How much more popular will C become? According to P.J. Plauger, involved with the early development of C at Bell Labs, C's popularity is just short of peaking. (See editor Craig LaGrow's interview with Plauger in ComputerVisions.)

A real highlight in this issue is our comparative analysis of 21 C compilers—nearly every C compiler now on the market. This is the most complete and objective (no re-run press releases here!) C review any magazine has ever published.

We established a four-man team of C experts who judged the compilers on objective technical criteria, comparing the compilers on the basis of four benchmark timing programs and a close analysis of their features. This consumer report kind of approach seems to be the fairest method of comparison possible, but I'm sure we'll receive a healthy amount of disagreement about the results. Let's just keep it down to idle threats—no physical harm, please!

Another important feature in this C theme issue is a special report from the chairman of the C Programming Language Standards Committee, Jim Brodie, on the history and current status of the C language standardization effort.

This issue also introduces a new column by Namir Shammas entitled CrossXthoughts. Look to this column for lively discussions on topics like programming techniques, algorithms, and practical solutions to complex programming problems.

I also want to announce our plans to sponsor a special seminar and workshop on the C programming language to be held in Cambridge, Mass., in September 1985. We're just setting up this event now and would appreciate any suggestions you might have. If you're interested in getting more information about this special event, please fill out the coupon on page 11 and send it in today. Additional details about the seminar will be released in upcoming issues.

This issue is a mini celebration of sorts—it's *COMPUTER LANGUAGE*'s half-year birthday. After only six issues, we're selling over 30,000 copies every month. Thanks for the great support.

Finally, remember *COMPUTER LANGUAGE* is not a cheerleader for any language. All Modula-2, Forth, COBOL, BASIC, etc., fanatics, you'll have your month too!

Cheers!

Carl Lan

Carl Landau
Publisher

Telecommunicate to COMPUTER LANGUAGE

COMPUTER LANGUAGE has established two bulletin board systems for you to upload and download text and binary programs, as well as to leave your own electronic Letter to the Editor. All the program listings referred to in every issue of the magazine will be available here.

In addition, *COMPUTER LANGUAGE* has its own Special Interest Group on CompuServe's national data base. After calling into your local CompuServe node, simply type "GO CLM" at any prompt and you'll be in our SIG.

To access our bulletin board, set your computer or terminal to the following parameters: 8 data bits, no parity, 1 stop bit, full duplex, and either 300 or 1200 baud. The telephone number is (415) 957-9370. After your modem makes the connection, type RETURN several times, and everything else is easy.

Both systems are open 24 hours per day, 7 days per week. Due to the heavy number of callers, please do not log into the system more than one time per day. Messages left on either system will be combined the following day.

COMPUTER
LANGUAGE

EDITOR
Craig LaGrow

MANAGING EDITOR
Regina Starr Ridley

TECHNICAL EDITOR
John Halamka

EDITORIAL ASSISTANT
Hugh Byrne, Lorilee Biernacki

CONTRIBUTING EDITORS
Doug Millison, Namir Shammas, Ken Takara, J. Edward Volkstorf Jr.

INDUSTRY NEWS CONSULTANT
Bruce Lynch

ADVERTISING SALES
Jan Dente

CIRCULATION COORDINATOR
Renato Sunico

ART DIRECTOR
Jeanne Schacht

COVER PHOTO
Dow Clement Photography

PRODUCTION ARTIST
Anne Doering

PRODUCTION
Barbara Luck, Steve Campbell, Kyle Houbolt

TECHNICAL CONSULTANT
Addison Sims

MARKETING CONSULTANT
Steve Rank

ACCOUNTING MANAGER
Lauren Kalkstein

PUBLISHER
Carl Landau

COMPUTER LANGUAGE *is published monthly by* COMPUTER LANGUAGE *Publishing Ltd., 131 Townsend St., San Francisco, CA 94107. (415) 957-9353.*

Advertising: For information on ad rates, deadlines, and placement, contact Carl Landau or Jan Dente at (415) 957-9353, or write to: COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, CA 94107.*

Editorial: Please address all letters and inquiries to: Craig LaGrow, Editor, COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, CA 94107.*

Subscriptions: Contact COMPUTER LANGUAGE*, Subscriptions Dept., 2443 Fillmore St., Suite 346, San Francisco, CA 94115. Single copy price: $2.95. Subscription prices: $24.95 per year (U.S.); $30.95 per year (Canada and Mexico). Subscription prices for outside the U.S., Canada, and Mexico: $36.95 (surface mail), $54.95 (air mail)—U.S. currency only. Please allow six weeks for new subscription service to begin.*

Postal information: Second-class postage rate is pending at San Francisco, CA and additional mailing offices.

Reprints: Copyright 1984 by COMPUTER LANGUAGE *Publishing Ltd. All rights reserved. Reproduction of material appearing in* COMPUTER LANGUAGE *is forbidden without written permission.*

Change of address: Please allow six weeks for change of address to take effect. POSTMASTER: Send change of address (Form 3579) to COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, CA 94107.*

COMPUTER LANGUAGE *is a registered trademark owned by the magazine's parent company, CL Publications. All material published in* COMPUTER LANGUAGE *is copyrighted © 1984 by CL Publications, Inc. All rights reserved.*

NEW from BORLAND!

TURBO TOOLBOX & TURBO TUTOR

Offer extended by popular demand!
Get your Borland Holiday Pack by March 1st, 1985.
TURBO TOOLBOX
POWER TOOLS FOR TURBO PASCAL
TURBO PASCAL
VERSION 2.0
BORLAND INTERNATIONAL
Turbo Tutor
A Turbo-Pascal Tutorial
BORLAND INTERNATIONAL
TURBO TOOLBOX

"TURBO is much better than the Pascal IBM sells."
Jerry Pournelle,
Byte, July 1984

"TURBO PASCAL appears to violate the laws of thermodynamics.
You won't find a comparable price/performance package anywhere. It is simply put, the best software deal to come along in a long time. If you have the slightest interest in Pascal . . . buy it."
Bruce Webster,
Softalk IBM: March 1984

BORLAND INTERNATIONAL GIFT PACK

ONLY $99.95

A SAVINGS OF $30!

What a gift for you and your friends! The extraordinary TURBO PASCAL compiler, together with the exciting new TURBO TOOLBOX and new TURBO TUTOR. All 3 manuals with disks for $99.95.

TURBO PASCAL Version 2.0 (reg. $49.95). The now classic program development environment still includes the FREE MICROCALC SPREAD SHEET. Commented source code on disk

- Optional 8087 support available for a small additional charge

NEW! TURBO TOOLBOX (reg. $49.95). A set of three fundamental utilities that work in conjunction with TURBO PASCAL. Includes:

- TURBO-ISAM FILES USING B+ TREES. Commented source code on disk
- QUIKSORT ON DISK. Commented source code on disk
- GINST (General Installation Program)

Provides those programs written in TURBO PASCAL with a terminal installation module just like TURBO'S!

- NOW INCLUDES FREE SAMPLE DATABASE . . . right on the disk! Just compile it, and it's ready to go to work for you. It's a great example of how to use TURBO TOOLBOX and, at the same time, it's a working piece of software you can use right away!

NEW! TURBO TUTOR (reg. $29.95). Teaches step by step how to use the TURBO PASCAL development environment—an ideal introduction for basic programmers. Commented source code for all program examples on disk.

30 DAY MONEY BACK GUARANTEE Available at your nearest software dealer.

For VISA and MASTERCARD order call toll free: **1-(800)-255-8008 1-(800)-742-1133**
(Lines open 24 hrs., 7 days a week) Dealer and Distributor inquiries welcome (408) 438-8400

CHOOSE ONE (please add $5.00 for handling and shipping U.S. orders)

___ All Three-Gift Pack	$ 99.95 + 5.00	**SPECIAL!**	___ Turbo Toolbox	$49.95 + 5.00
___ All Three & 8087	139.95 + 5.00	**SPECIAL!**	___ Turbo Tutor	29.95 + 5.00
___ Turbo Pascal 2.0	49.95 + 5.00		___ Turbo 8087	89.95 + 5.00

Check__________ Money Order__________ VISA__________ MasterCard__________
Card #: ______________________________ Exp. date: ______________ Shipped UPS
My system is: 8 bit ___ 16 bit ___
Operating System: CP/M 80 ___ CP/M 86 ___ MS DOS ___ PC DOS ___
Computer: ______________________ Disk Format: ______________________
Please be sure model number & format are correct.
NAME: ______________________________
ADDRESS: ______________________________
CITY/STATE/ZIP: ______________________________
TELEPHONE: ______________________________

California residents add 6% sales tax. Outside U.S.A. add $15.00 (if outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars). Sorry, no C.O.D. or Purchase Orders.

20

BORLAND
INTERNATIONAL

4113 Scotts Valley Drive
Scotts Valley, California 95066
TELEX: 172373

CIRCLE 6 ON READER SERVICE CARD

AZTEC The Most Powerful C

for the IBM AT • MACINTOSH • MS DOS • CP/M-80 • ROM APPLICATIONS IBM PC/XT • APPLE // • CP/M-86 • TRSDOS • CROSS DEVELOPMENT

Why Professionals Choose Aztec C

AZTEC C compilers generate fast, compact code. AZTEC C is a sophisticated development system with assemblers, debuggers, linkers, editors, utilities and extensive run time libraries. AZTEC C is documented in detail. AZTEC C is the most accurate and portable implementation of C for microcomputers. AZTEC C supports specialized professional needs such as cross development and ROM code development. MANX provides qualified technical support.

AZTEC C86/PRO

— for the IBM AT and PC/XT

AZTEC C86/PRO provides the power, portability, and professional features you need to develop sophisticated software for PC DOS, MS DOS AND CP/M-86 based microsystems. The system also supports the generation of ROM based software for 8088/8086, 80186, and 80286 processors. Options exist to cross develop ROM code for 65xx, 8080, 8085, and Z80 processors. Cross development systems are also available that target most micro computers. Call for information on AZTEC C86/PRO support for XENIX and TOPVIEW.

POWERFUL — AZTEC C86/PRO 3.2 outperforms Lattice 2.1 on the DHRYSTONE benchmark 2 to 1 for speed (17.8 secs vs 37.1) while using 65% less memory (5.8k vs 14k). The AZTEC C86/PRO system also compiles in 10% to 60% less time and supports fast, high volume I/O.

PORTABLE — MANX Software Systems provides **real** portability with a family of compatible AZTEC C software development systems for PC DOS, MS DOS, CP/M-86, Macintosh, CP/M-80, APPLE //+, //e, and //c (NIBBLE - 4 apple rating), TRSDOS (80-MICRO - 5 star rating), and Commodore C64 (the C64 system is only available as a cross compiler - call for details). AZTEC C86/PRO is compatible with UNIX and XENIX.

PROFESSIONAL — For professional features AZTEC C86/PRO is unparalleled.

- Full C Compiler (8088/8086 - 80186 - 80286)
- Macro Assembler for 8088/8086/80186/80206
- Linkage Editor with ROM support and overlays
- Run Time Libraries - object libraries + source DOS 1.x; DOS 2.x; DOS 3.x; screen I/O; Graphics; UNIX I/O; STRING; simulated float; 8087 support; MATH; ROM; CP/M-86
- Selection of 8088/8086, 80186, or 80286 code generation to guarantee best choice for performance and compatibility
- Utility to convert AZTEC object code or libraries to Microsoft format. (Assembly + conversion takes less than half the time as Microsoft's MASM to produce MS object)
- Large memory models and sophisticated memory management
- Support products for graphics, DB, Screen, & ...
- ROMable code + ROM support + separate code and data + INTEL Hex Converter
- Symbolic Debugger & Other Utilities
- Full Screen Editor (like Vi)
- CROSS Compilers are available to APPLE //, Macintosh, CP/M-80, TRSDOS, COMMODORE C64, and ROM based 65xx, and 8080/8085/Z80
- Detailed Documentation

AZTEC C86/PRO-AT (configured for IBM AT - options for 8088/8086)	$500
AZTEC C86/PRO-PC/XT (configured for IBM PC/XT - options for 80186/80286)	$500

AZTEC C86/BAS includes C compiler (small model only), 8086 MACRO assembler, overlay linker, UNIX, MATH, SCREEN, and GRAPHICS libraries, debugger, and editor.

AZTEC C86/BAS	$199
AZTEC C86/BAS (CP/M-86)	$199
AZTEC C86/BAS (DOS + CP/M-86)	$299
UPGRADE to AZTEC C86/PRO	$310
C-TREE Database with source	$399
C-TREE Database (object)	$149

CROSS COMPILERS

Cross Compilers for ROM, MS DOS, PC DOS, or CP/M-86 applications.

VAX - > 8086/80xxx cross	$5000
PDP-11 - > 8086/80xxx cross	$2000

Cross Compilers with PC DOS or CP/M-86 hosts are $750 for the first target and $500 for each additional target. Targets: 65xx; CP/M-80; C64; 8080/8085/Z80; Macintosh; TRSDOS; 8086/8088/80186/80286; APPLE //.

AZTEC C68K

— for the Macintosh

For power, portability, and professional features AZTEC C68K-c is the finest C software development system available for the Macintosh.

The AZTEC C68K-c system includes a 68000 macro assembler, a linkage editor, a source editor, a mouse based editor, a SHELL development environment, a library of UNIX I/O and utility routines, full access and support of the Macintosh TOOLBOX routines, debugging aides, utilities, make, diff, grep, TTY simulator with upload & download (source supplied), a RAM disk (for 512K Mac), a resource maker, and a **no royalty** license agreement. Programming examples ar included. (Over 600 pages of documentation).

AZTEC C68K-c requires a 128K Macintosh, and two disk drives (frugal developers can make do with one drive). AZTEC C68K supports the 512K Macintosh and hard disks.

AZTEC C68K-c (commercial system)	$500
AZTEC C68K-p (personal system)	$199
AZTEC C68K-p to AZTEC C68K-c upgrade	$310
Mac C-tree database	$149
Mac C-tree database with source	$399
Lisa Kit (Pascal to AZTEC C68k object converter)	$ 99

AZTEC C65

— for the APPLE //

"...The AZTEC C-system is one of the finest software packages I have seen..." NIBBLE review, July 1984.

The only commercial C development system available that runs native on the APPLE II+, IIc, and IIe, the AZTEC C65 development system includes a full floating point C compiler compatible with UNIX C and other MANX AZTEC C compilers, a 6502 relocating assembler, a linkage editor, a library utility, a SHELL development environment, a full screen editor, UNIX I/O and utility subroutines, simple graphics, and screen functions.

AZTEC C65 (Apple DOS 3.3)	$199
AZTEC C65/PRO (Apple DOS + ProDos) (call for availability)	$350

AZTEC C II/PRO

— for CP/M-80

The first member of the AZTEC C family was the CP/M-80 AZTEC C compiler. It is "the standard" compiler for development on CP/M-80. The system includes the AZTEC C II C compiler, an 8080 assembler, a linkage editor, an object librarian, a full library of UNIX I/O and utility routines, CP/M-80 run time routines, the SMALL library (creates modules less than 3K in size), the fast linker for reduced development times, the ROM library, RMAC and M80 support, library source, support for DRI's SID/ZSID symbolic debugger, and more.

AZTEC C II/PRO	$349
AZTEC CII/BAS	$199
C-TREE Database with source	$399
C-TREE Database in AZTEC object form	$149

AZTEC C80

— for TRSDOS (Radio Shack Model III & 4)

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen." 80-Micro, December, 1984, John B. Harrell III

This sytem has most of the features of AZTEC C II for CP/M. It is perhaps the best software development system for the Radio Shack Model III and IV.

AZTEC C80 model 3 (no floating point)	$149
AZTEC C80 model 4 (full)	$199
AZTEC C80/PRO (full for model 3 and 4)	$299

To order or for information call:

800-221-0440

(201) 530-7997 (NJ and outside U.S.A.). Or write: MANX SOFTWARE SYSTEMS, P.O. Box 55, Shrewsbury, N.J. 07701.

MANX

SHIPPING INFORMATION - Standard U.S. shipment is UPS ground (no fee). In the U.S. one day shipment is $20, two days is $10. Canadian shipment is $10. Two days shipment outside the U.S. is by courier and is freight collect.

TRS 80 RADIO SHACK TRS DOS is a trademark of TANDY.
APPLE DOS MACINTOSH is a trademark of APPLE.

For Technical Support
(Bug Busters) call: 201-530-6557

CIRCLE 69 ON READER SERVICE CARD

FEEDBACK

A C limerick

Dear Editor:

C is one of the most versatile languages available today. Its ability to add by function allows you to create large applications. Yet for operating systems or process control, C can "get small" while still maintaining good structure.

As a mildly typed language, C impresses some restrictions on the programmer but still allows freedom for special situations. I thought your readers might enjoy a little program I wrote for nonprogramming managers who insist on easily readable code (Listing 1).

Mike Rejsa
Minneapolis, Minn.

```
/*  A limerick for C programmers  */
/*  by Mike Rejsa                 */
#define a
#define and(c) bdos(6, c)
#define created
#define Dee =
#define had =
#define it --
#define thought puts
#define with -
int *fame = 128, *give = 128, *name = 128;
int *Bea = 130, *looking = 130, *to = 130;
int language = 120;
main()
{
      /* A lady programmer named */
      *Bea Dee
      created a language
      with 'C';
      while (--*looking)
      {
              for (*fame had *to; *give; it a *name)
                      and( ' ' );
              thought( "Is it Bea, C, or Dee?" );
      }
}
```

Listing 1.

Correct Ada figure

Dear Editor:

For those people who want a correct copy of Figure 2 of my article "Learn to Think in Ada" (Nov. 1984, pp. 47-49), here is a copy (Listing 2). Please apologize to our readers for me. I can imagine how frustrating it must be to try to figure out how a program works when it isn't all there.

Do-While Jones
Ridgecrest, Calif.

```
with CONSOLE_IO; use CONSOLE_IO;
procedure Solve_Exercise_3_4_B is
  I, J : integer;
  TEST_RESULT, I_IS_ODD, J_IS_EVEN : boolean;
begin
  put("Please enter an integer ""I"" "); get(I); new_line;
  put("Now enter another integer ""J"" "); get(J); new_line;
  if I mod 2 = 1
    then I_IS_ODD := TRUE;
    else I_IS_ODD := FALSE;
  end if;
  if J mod 2 = 0
    then J_IS_EVEN := TRUE;
    else J_IS_EVEN := FALSE;
  end if;
  TEST_RESULT := I_IS_ODD and J_IS_EVEN;
  put("The test turned out "); put(TEST_RESULT); new_line;
end Solve_Exercise_3_4_B;
```

Listing 2.

What was it, exactly?

As many readers have pointed out, the listings in Joe Celko's "What Day is It, Exactly" (Dec. 1984, pp. 47-50) contained some errors. Here are corrections, by listing and line number:

- *Listing 1/line 4: change MOD 400 to MOD 100*
- *Listing 2/line 1: change Validate to ValidDate*
  line 4: change MinSystemYear to MaxSystemYear
  line 11: change MonthSuze to MonthSize
  line 12: change second MinSystemYear to MaxSystemYear
  line 13: change Validdate to ValidDate
- *Listing 3/line 2: add */ to end of line*
  line 8: change INIT90 to INIT (0
  line 9: change month > 1 to month > 2

```
IF (month=2) THEN Jul2:= Jul2 + 1 ELSE NULL;
IF ((month>2) AND (month<9)) THEN Jul2:= Jul2 - 1 ELSE NULL;
```

Listing 3.

```
IF LeapYear(year) THEN X:=1 ELSE X:=2
Julian:= Truncate(275 * month / 9) - X * Truncate((month+9) / 12)
Julian:= Julian + day - 30;
```

Listing 4.

line 12: change day: to day;
- *Listing 4/line 2: change * to */*

line 7: change 30.475 to 30.4375

line 12: change month > 1 to month > 2
- *Listing 5/line 8: change CenturyTable[1:12] to CenturyTable[1:5]*

Line 26: change : to ; at the end of line
- *Listing6/line 12: change EF to IF.*

Calendar modifications

Dear Editor:

Lines 8-11 of Listing 4 in Joe Celko's "What Day is It, Exactly?" are wrong. They might be replaced with the two lines in Listing 3.

An easier approach to the Commercial Date is shown in Listing 4.

F. Barry Mulligan
Atlanta, Ga.

Leap year rule

Dear Editor:

I feel I should correct an inaccuracy in Joe Celko's "What Day is It, Exactly?" The inaccuracy has to do with leap year calculations.

First, in the text of Celko's article he states, " . . . every year divisible by four is a leap year . . . there is also a 400-year cycle . . . The four-year cycle exists because the number of days in a year is 365 and almost one-quarter days. Leap year takes care of the one-quarter day, and the 400-year rule takes care of the 'almost.' "

The actual rule is different. The year is a little more than 11 min short of 365 days and one-quarter. This amounts to being a day ahead after about 128 years or about 3 days ahead in 400 years. The rule is: if the year is divisible by 4, it IS a leap year, UNLESS, if the year is divisible by 400, it IS a leap year. Hence, the year 2000 is a leap year.

Daniel Efron
St. Louis Park, Minn.

Author Joe Celko comments: Where 365.25 days per year appears, you can substitute the more accurate 365.2422 days per year if the software will not have rounding problems in the calculations.

For your IBM/PC

mbp COBOL: 4 times faster, and now with SORT & CHAIN.

$750.

mbp COBOL can be summed up in one word: fast.

Because it generates native machine language object code, the mbp COBOL Compiler executes IBM/PC* programs *at least* 4 times faster (see chart).

GIBSON MIX Benchmark Results
Calculated S-Profile
(Representative COBOL statement mix)

Execution time ratio

mbp COBOL	Level II** COBOL	R-M*** COBOL	Microsoft**** COBOL
1.00	4.08	5.98	6.18

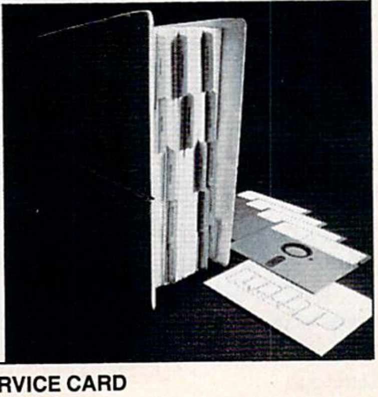
128K system with hard disk required. *IBM/PC is an IBM TM; **Level II is a Micro Focus TM; ***A Ryan-McFarland TM; ****A Microsoft TM.

Fast also describes our **new SORT,** which can sort four-thousand 128-byte records in less than 30 seconds. A callable subroutine or stand-alone, 9 SORT control fields can be specified. And our **new CHAIN** is both fast and secure, conveniently transferring control from one program to another, passing 255 parameters. Plus, **new extensions** to ACCEPT & DISPLAY verbs give better, faster interactive programming.

The complete COBOL. An Interactive Symbolic Debug Package included standard; Multi-Keyed ISAM Structure; listing options allow source & object code, map & cross-reference checking; GSA Certification to ANSI '74 Level II; mbp has it all.

It's no surprise companies like Bechtel, Chase, Citicorp, Connecticut Mutual, and Sikorsky choose mbp COBOL; make it your choice, too. mbp is available at Vanpak Software Centers, or direct. For complete information, write **mbp Software & Systems Technology, Inc.,** 7700 Edgewater Drive, Suite 360, Oakland, CA 94621, or phone 415/632-1555 —today.

mbp®



CIRCLE 39 ON READER SERVICE CARD

```
PROGRAM Math
PARAMETER (a=1.43576, b=5.23417)
total = 0.0
DO 10 i = 1, 2500
  c = EXP( LOG( (SQRT(b))**2 ) ) * a/b
  d = ATAN( SIN(c)/COS(c) ) /a
  total = total + d
10 CONTINUE
WRITE(*,20) total
20 FORMAT(1X,'Total is: ',F11.5)
STOP
END
```

```
PROGRAM Overhead
PRINT *, 'Finished'
STOP
END
```

Listing 5.

FORTRAN math program

Dear Editor:

Your software review of the Digital Research FORTRAN-77 compiler (Nov. 1984, pp. 66-71) was worthwhile but it lacked any critical comments. Being interested in such a compiler, I visited a friend who recently purchased the DRI product and is running it on an NEC/APC operating under CP/M 86. We entered the two programs in Listing 5.

The resulting statistics for program "overhead" are provided: (most of the run-time is consumed in reading the executable image into RAM)

Compilation time: 41 sec
Link time: 5 min 24 sec
Run time: 16 sec
Executable file size: 84K

The math program was compiled without options and then linked with the HARD8087 option (requires 8087 coprocessor) and also with the SIM8087 option (software mathematics), and these two executable images were run from a RAMdisk. The following statistics are provided:

- On the 8086 math program, an execution time of 15 min 15 sec, and a result of 2499.99976
- On the 8087 chip, an execution time of 15.7 sec, and a result of 2499.99976.

Note that the correct result is 2500.

Kenneth C. Beaudrie
Denver, Colo.

ANNOUNCING . . . COMPUTER LANGUAGE'S

C Seminar/Workshop

Cambridge, MASS
September 1985

Plans are being set now for COMPUTER LANGUAGE's C Seminar. The 2½ day event will be held in the fall of 1985 in beautiful Cambridge.

Details concerning the topics, speakers and dates will be announced soon. This seminar will be the most comprehensive and practical session ever held about the C programming language.

Become involved in this event from the start by filling out this coupon today:

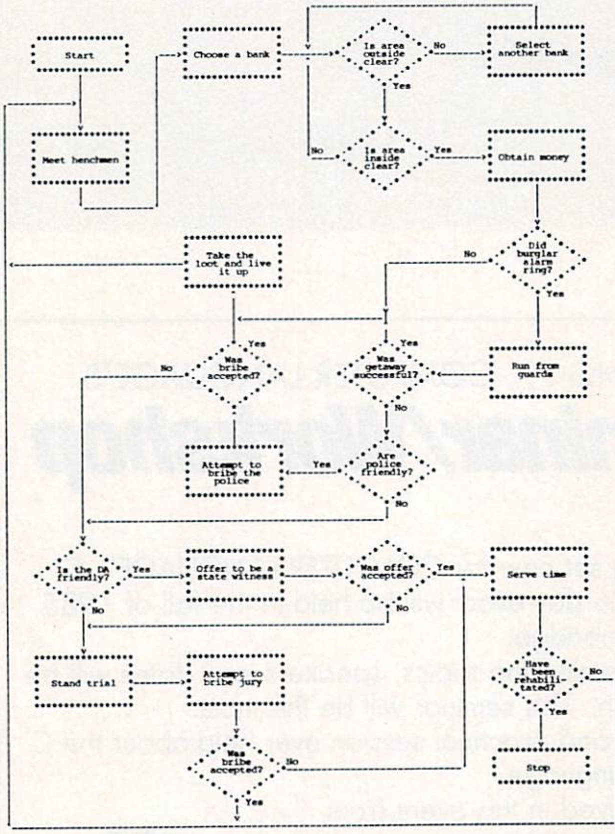
☐ Please send me more information regarding the seminar and Early Bird Discounts.
☐ I would like to present a paper about:
☐ I suggest the following speakers:
☐ I would like to see these topics covered:

Name
Company
Address
City, State, Zip
Phone

Send to: COMPUTER LANGUAGE Seminar
131 Townsend Street • San Francisco, CA 94107

WARNING:

Do NOT read this flowchart:


Start
Choose a bank
Is area outside clear?
No
Select another bank
Yes
Meet henchmen
No
Is area inside clear?
Yes
Obtain money
Take the loot and live it up
No
Did burglar alarm ring?
Yes
Yes
No
Was bribe accepted?
Yes
Was getaway successful?
Run from guards
No
Attempt to bribe the police
Yes
Are police friendly?
No
Is the DA friendly?
Yes
Offer to be a state witness
Was offer accepted?
Yes
Serve time
No
No
Stand trial
Attempt to bribe jury
Have I been rehabilitated?
No
Yes
Was bribe accepted?
No
Stop
Yes

Unless you have time to spare ... in extreme cases a few people have found they had thirty years to spare.

You read the flowchart anyway. Why? Because flowcharts are a powerful graphic way of communicating ideas. The big problem is producing them: they take a lot of time to do well and are difficult to revise or correct without starting over from scratch.

The flowchart above was produced using **EasyFlow,** a computer aided flowchart generation tool. You decide how the flowchart is to be laid out and describe the flowchart to EasyFlow using a simple command language. EasyFlow then does the hard part of actually producing and printing the flowchart.

EasyFlow is a well designed, thoroughly tested and comprehensively documented package.

- Fast. EasyFlow is written in assembly language for speed. It produces a typical flowchart in 12 seconds.
- Easy. The EasyFlow flowchart description language is straightforward and easy to learn.
- Printers. EasyFlow works with all printers. Extensive printer control facilities allow you to fine-tune EasyFlow to produce the best possible flowcharts on your particular printer.
- Shapes. EasyFlow comes with eighteen standard flowcharting shapes. User defined shapes can be easily added.
- Lines. User definable line-drawing characters allow EasyFlow to adapt to personal preference and printers with graphic line characters.
- Manual. Complete, comprehensible and over 100 pages long. Also included is a quick reference card, a planning grid and ten demo flowcharts (demo flowcharts supplied on disk as well).

$49.95 (U.S. funds) Check, money order, Visa or company P.O.

EasyFlow is available for MS-DOS/PC-DOS machines (96K or greater) on IBM PC format 5″ diskettes. Available for CP/M-80 machines (48K or greater) on 8″ SSSD and most soft sectored 5″ formats.

HavenTree Software Limited
R.R. #1, Suite 100
Seeley's Bay, Ontario
Canada, K0H 2N0
(613) 542-7270

CIRCLE 41 ON READER SERVICE CARD

MULTI-BASIC

The Compatible BASIC Compiler from Alcor

(Supported Features Chart)

	MBASIC	MBASIC Compiler	CBASIC	CBASIC Compiler	Multi-Basic
IF ELSE/FOR NEXT	•	•	•	•	•
WHILE WEND	•	•	•	•	•
OPEN/CLOSE	•	•	•	•	•
CREATE/DELETE			•	•	•
PRINT/PRINT USING	•	•	•	•	•
LPRINT/LPRINT USING	•	•			•
PRINT @	•	•			•
INPUT/LINE INPUT	•	•	•	•	•
INPUT #/LINE INPUT #	•	•			•
READ #/READ # LINE			•	•	•
LOC/LOF/EOF/ERROR	•				•
FIELD/GET/PUT	•	•			•
RSET/LSET/ERASE	•	•			•
MKD$/MKI$/MKS$	•	•			•
CVI/CVS/CVD/SPC	•	•			•
READ/DATA/RESTORE	•	•	•	•	•
TRON/TROFF	•				•
DEF FN/RANDOM/RND	•	•	•	•	•
DEF USR/SWAP/WAIT	•	•			•
CALL (ASSEMBLY LANG)	•				•
DEFSTR/DBL/SNG/INT	•	•			•
DOUBLE/REAL			•	•	•
INTEGER/STRING			•	•	•
GOTO/GOSUB	•	•	•	•	•
ON ERROR GOTO	•			•	•
RESUME/RESUME NEXT	•				•
ERL/ERR	•			•	•
ON number GOTO/GOSUB	•	•	•	•	•
NAME/RENAME	•	•	•	•	•
PEEK/POKE/INP/OUT	•	•	•	•	•
SYSTEM/SOUND	•				•
SADD/MATCH/UCASE$			•	•	•
VAL/TAB/STR$/VARPTR	•	•	•	•	•
SIN/COS/TAN/ATN	•	•	•	•	•
LOG/EXP/ABS/SQR	•	•	•	•	•
COMMAND$/IF END			•	•	•
INKEY$/INPUT$	•	•			•
TIME$/DATE$/HEX$/OCT$	•	•			•
STRING$/SPACE$	•	•			•
LEFT$/RIGHT$/MID$	•	•	•	•	•
CHR$/ASC/LEN/SGN	•	•	•	•	•
OPTION BASE	•				•
ROW/POS/LPOS	•				•
DIM/MEM/FRE	•	•	•	•	•
MOD/MFRE				•	•
GET/PUT (CHARACTER)			•	•	•
KILL/CLS/CLEAR/INSTR	•	•			•
INT/FIX/CINT	•	•			•
CSNG/CDBL	•	•			•
CHAIN	•		•	•	•
CONSTAT%/CONCHAR%			•	•	•
CONSOLE/LPRINTER			•	•	•
BINARY RANDOM FILES	•	•			•
ASCII RANDOM FILES			•	•	•
255 CHARACTER NAMES				•	•
REDIMENSIONED ARRAYS	•		•	•	•
UNLIMITED STRING SIZE				•	•
MULTI-LINE FUNCTIONS			•	•	•
PROCEDURES/RECURSION					•
FUNCTION TRACING					•
LOCAL VARIABLES			•	•	•
NESTED FUNCTIONS					•
OPTIONAL LINE NUMBERS			•	•	•
DESCRIPTIVE LABELS			•	•	•
SINGLE PRECISION	•	•			•
DOUBLE PRECISION	•	•	•	•	•
LINK TO PASCAL & C					•

For TRS80 models I, II, III, 4, 12, or 2000 using TRSDOS, CP/M, or MSDOS and for IBM PC using PCDOS
Multi-Basic, Pascal, or C $139
Add shipping $6.00 USA, $28 Overseas

ALCOR Systems

13534 Preston Road, Suite 365
Dallas, Texas 75240
(214) 238-8554

Multi-Basic is a trademark of Alcor Systems
TRSDOS, TRS80 is a registered trademark of Tandy Corporation
CP/M, CBASIC are trademarks of Digital Research
MSDOS, MBASIC are trademarks of Microsoft

CIRCLE 1 ON READER SERVICE CARD

CROSSXTHOUGHTS

Pseudocode syntax

By Namir Shammas

Welcome to CrossXthoughts! It is a new column for all programmers and a platform for sharing algorithms and programming development techniques and solutions.

We will cover a wide variety of subjects, such as data management techniques, interpreters, graphics, and artificial intelligence. I will be using the *COMPUTER LANGUAGE* Bulletin Board Service (415 957-9370) and CompuServe to interact with readers (but ordinary mail is not completely ruled out!). These media will be used for topic selections, follow-up discussion on published columns, questions asked by readers, and sharing expert opinions and experiences.

In addition, the column will assist readers with similar interests in contacting each other. This will allow further discussion and exchange and published follow-up. I will maintain SIG lists containing names, addresses, phone numbers, etc. They will be posted on the bulletin boards and also made available through *COMPUTER LANGUAGE* by written requests. Putting your name on a particular SIG list is done at your request, by writing or through the BBS and CompuServe. The column will also carry out mini-contests awarding books, software packages, and subscriptions in the magazine.

To make this column a crossroad for programmers who use different languages, I will be using pseudocode listings frequently. Pseudocode is also this issue's topic.

To insure good readability and a degree of consistency I am suggesting a syntax for the Programmers' Pseudo Listing (PPL). I have loaded it on a BBS file called NILE, which contains an application program written in Turbo Pascal and the documentation for using it. It checks the PPL syntax and helps in top-down software design process, allowing development from general to specific.

Additional advantages gained from using the PPL are:

- During software development it is easier and faster to alter and update text than it is to change flowcharts.
- The PPL offers programmers flexibility in sharing information. One can generalize certain code segments to either spare the other party irrelevant code or deliberately hide detailed algorithms to protect programming tips and tricks.
- To assist top-down software design.
- To assist applications development in more than one language or from one to another.

The implementation of a standard PPL, just like any language, is subject to certain syntax and rules. They are of course much more flexible than those for formal languages. Deviating from them is by no means a sacrilege. The general rules are:

- **Data types and identifiers.** This is the area where languages differ greatly. The PPL will leave the choice of how to present data structures up to the programmer.
- **Labels.** The aim of structured languages is to move away from using labels and *GOTO*s. It is worth pointing out that Modula-2 has not implemented the *GOTO* statement at all! By contrast, Ada and C allow using the *GOTO* especially for exiting deeply nested loops. Therefore, labeled *GOTO*s are allowed. This should benefit those who program in BASIC and FORTRAN. However, I must say—following the example of nonsmoker signs—"Thank you for not *GOTO*ing!"
- **Loops.** There are two basic types of loops. The first has a loop counter with starting and ending values—the code in the loop is repeated a fixed number of times. This is the famous *FOR-DO* loop in Pascal and Modula-2 or the *FOR-NEXT* in BASIC.

The second loop type would execute the enclosed code depending on the success of a conditional test. When the test is located at the beginning of the loop, we have the *WHILE-DO*, as in Pascal, Modula-2, C, and FORTRAN 77. When the test is carried out at the end of the loop, forcing execution of the code at least once, we have the *REPEAT-UNTIL* effect. The test can also be carried out anywhere inside the loop in an attempt to exit. This is equivalent to the *LOOP-EXIT* in Ada and Modula-2.

From the preceding we can see that the different loop types can be presented basically as shown in Listing 1. The *INITIALIZE* will remind the programmer of any loop initializations; I suggest its presence be mandatory. The word "None" should be used when no action is needed. The *LOOP* keyword is followed by an optional loop name. The latter is useful to keep track of exiting nested loops (actually, it is an Ada feature).

The loop *BEGIN* may contain information to indicate the use of a counter or a conditional test to simulate the *WHILE-DO* construct. The conditional test can be inserted inside the loop code or at the end. The latter will simulate the *REPEAT-*

```
General loop construct

INITIALIZE  <optional set-up>
LOOP <optional name>
BEGIN <no text to simulate REPEAT-UNTIL> or
      <declare loop counter> or
      <Optional test here to EXIT <optional name> >

   loop code with optional test to EXIT <optional name>

   Optional test here to EXIT <optional name>

END LOOP
TERMINATE resolve any pending operations
```

Listing 1.

UNTIL construct. The *END LOOP* token need not be followed by a loop name since we expect loops to be logically nested. The *TERMINATE* keyword is used to indicate the action taken to resolve pending data treatment after the loop ends.

■ **Conditional branching.** Inspired by languages like C, Pascal, Modula-2 and Ada, I use two types of conditional branching constructs:

1. The well-known *IF-THEN* construct with *ELSE* and *ELSEIF* clauses for more elaborate decision-making. Listing 2 shows the two alternatives when using the *IF-THEN* clauses.

2. The *CASE* test is applied when an identifier can be used as a switch to select the appropriate course of action. Listing 3 shows the construct for *CASE*. The *WHEN* and *OTHERWISE* keywords will indicate the beginning of a new *CASE* option and the end of a previous one, except for the first *WHEN*.

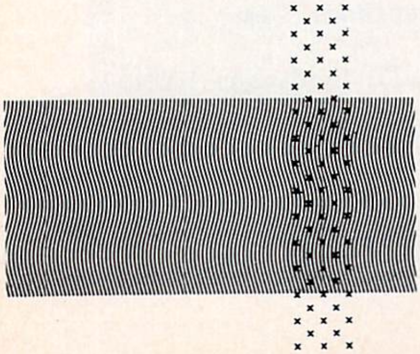
```
IF-THEN constructs

IF <Test is true>
THEN
            Outcome # 1
ELSE
            Outcome # 2
END IF

and

IF <Test 1  is true>
THEN
      Outcome # 1
ELSEIF <Test 2 is true>
THEN
      Outcome # 2
ELSEIF <Test 3 is true>
THEN
      Outcome # 3
ELSE
      Outcome # 4
END IF
```

Listing 2.



■ **Procedures and functions.** Due to the use of plain English words to describe the action taken, we must be able to point out specific calls to procedures and functions. I suggest that the procedure names be enclosed in brackets and function names in braces. The following example shows how a call to a procedure *SOLVE* is documented:

```
Set up simulataneous equation and
      [SOLVE]
```

Another example is a call to a logical function *DiskFull*:

```
IF {DiskFull} THEN warn the user that
      no space is left
```

Since the PPL may be involved in early software development stages, the procedure and function declarations follow the main code portion. This allows for a top-down software design. Thus, procedures and functions that are called more often would be located further down in the listing. To declare the code for a procedure or function, it is sufficient to use the keywords *PROCEDURE* or *FUNCTION* followed by a name.

■ **Input/Output.** This is another aspect of programming where different languages as well as dialects implement differently. We will adopt the following commands:

DISPLAY <list> for screen output
INPUT <list> for keyboard input
PRINT <list> to send output to a printer.

File I/O involves reading and writing of data records. The first command needed by all file I/O is to open it:

```
OPEN <filename>, <buffer name or
      number>, <mode>
```

where the I/O mode is one of the following:

INPUT for data input
OUTPUT for data output
APPEND to append output data to the existing file
RANDOM for random data read and write.

Its counterpart is *CLOSE <buffer name or number>* to close the file.

For sequential I/O we have *READ <buffer>, <data>* to input data, *WRITE <buffer>, <data>* to output. For random I/O we have *READ <buffer>, <record #>, <data>* and *WRITE <buffer>, <record #>, <data>* to read and write, respectively. The *SEEK <buffer>, <record>* is used to position record pointers to a specific location.

Port communications is carried out in a similar way to file I/O except the modes allowed are, obviously, *INPUT* and *OUTPUT*.

■ **Incrementing, decrementing and scaling variables.** These operations are used extensively in all languages. Some, like C, offer a shorthand way of stating them. I am adopting the C notation. This is useful in shortening code lines since long variable names would most likely be used for clarity. To increment a variable:

Variable += Added number

which is equivalent to

Variable = Variable + Added number

and to decrement a variable:

Variable −= Subtracted number

Similarly, scaling a variable by multiplication or division, as in

Variable = Variable * Factor

and

Variable = Variable / Factor

become (Variable *= Factor) and (Vari-

```
CASE construct

CASE <identifier>

      WHEN <value 1 is ture> => Outcome # 1
      WHEN <value 2 is ture> => Outcome # 2
      .........
      OTHERWISE => Outcome # X

END CASE
```

Listing 3.

able /= Factor), respectively.

■ **Remarks.** For those who want to formalize remarks and comments, borrowing from Ada's syntax is useful. The remark is expressed by at least two consecutive dashes, as in:

```
-- This is a remark
----------- This one too!
------- and so is this ----
```

Listing 4 shows pseudocode for solving a single nonlinear equation using the Newton-Raphson method. The PPL shown reflects a rather detailed code revealing very specific algorithms.

Listing 5 shows the code for a regression program that will read data from a file and perform a best-fit regression. This is done by subjecting the data read to a combination of mathematical transformations. Each of the latter represents a different regression model.

The code shown will process the data through four models. The slope, intercept and correlation coefficient is calculated for each model. An insertion sort is performed as results are obtained for each model. This allows for the printing of the curve fitting information sorted from best to worst model. The PPL listing reflects the case where a programmer is avoiding detailed coding in certain areas.

Do you have any comments or suggestions concerning PPL? Does the idea of a consistent type of pseudocode appeal to you? Is it too restrictive? Would you like to participate in a SIG-PPL group for further refinement of PPL? Let me hear from you. Please write to me in care of *COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, Calif. 94107 or contact me through the BBS or CompuServe.

In the next issue I will discuss interactive parsers and interpreters for mathematical equations (i.e., the heart of electronic spreadsheets). If you have any code or algorithms to share, questions to ask, or would like to enrole in the SIG list, drop me a line. Remember that the SIGs are there to help exchange information and, who knows, perhaps find your next software coauthor!

```
PPL listing for solving a single nonlinear equation
using the Newton-Raphson method

PROGRAM Root

Version 1.0   June 10, 1984
LAST UPDATE :  06-15-84       12:22:23

INPUT Guess, Accuracy, Maximum

  INITIALIZE   Set Counter = 0
               Set Goflag True
  LOOP
-- The token below is not followed by anything as part of
-- simulating the REPEAT-UNTIL loop
  BEGIN

       IF   abs(Guess) > 1    THEN    Increment = 0.01 * Guess
                              ELSE    Increment = 0.01
       END IF

       Delta_X = 2 * Increment * {F(Guess)} /
                ({F(Guess + Increment)} - {F(Guess + Increment)}

       Guess -= Delta_X

       Counter += 1
       IF   Counter > Maximum   THEN   Goflag is False   END IF


       When (abs(Delta_X) <= Accuracy) OR (Goflag is False)´EXIT

   END LOOP
  TERMINATE   None
  IF   Goflag is True   THEN
                          DISPLAY Guess and Counter
                        ELSE
                          DISPLAY Message for divergence
  END IF

END PROGRAM -- This is optional
-------------------------------------------------------------
FUNCTION {F(X)}
-- User function

    F = EXP(X) - 3 * X * X

END

END PPL

------------------------------------------------------------------
```

Listing 4.

PRENTICE-HALL:

THE C/UNIX*
Hotline

C TITLES

1. Programming in C with a Bit of UNIX*, F. Richard Moore. A first glimpse of C for the computer novice. 1985, 388 pp., paper (73009-3) $16.95; cloth (73010-1) $22.95.

2. Learning to Program in C, Thomas Plum. Plum walks programmers through the world of C. 1984, paper (52784-6) $25.00; cloth (52785-3) $32.95.

3. C Programming Guidelines, Thomas Plum. Put yourself on solid ground with this clear explanation of the standards. 1984, 145 pp., paper (10999-1) $26.00.

4. The C Programmer's Handbook, AT&T Bell Laboratories, M. I. Bolsky. Destined to be the C programmer's constant companion. 1985, 96 pp., spiral (11007-2) $14.95.

5. The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie. The "Bible"—a must for C programmers. 1978, 228 pp., cloth (11016-3) $22.95.

6. C: A Reference Manual, Samuel P. Harbison and Guy L. Steele. The latest details of C—right at your fingertips! 1984, paper (11000-7) $19.95; cloth (11001-5) $24.95.

7. The C Puzzle Book, Alan R. Feuer. Challenge your mind with these C brain teasers. 1982, paper (10992-6) $14.95; cloth (10993-4) $18.95.

8. Comparing and Assessing Programming Languages—Ada, C, and Pascal, Alan R. Feuer and Narain Gehani. Which language suits you best? Feuer and Gehani spell it out! 1984, 256 pp., paper (15484-9) $16.95; cloth (15485-6) $24.95.

UNIX TITLES

1. The UNIX* Programming Environment, Puts two UNIX experts right at your side. 1984, paper (93768-0) $19.95; cloth (93769-6) $26.95. Brian Kernighan and Rob Pike

2. A UNIX* Primer, Ann Nicols Lomuto and Nico Lomuto. Lays out UNIX by cutting through common-problems. 1983, 240 pp., paper (93773-0) $20.00; cloth (93888-6) $21.95.

3. UNIX* for People, Peter Birns, Patrick B. Brown, John C. Muster. Learn the UNIX language in your language! 1984, 544 pp., paper (93744-1) $19.95; cloth (93745-8) 29.95.

ORDER YOUR FREE 15-DAY TRIAL COPIES TODAY!

By phone: dial (201) 767-9520 **or write** to Steven T. Landis, Prentice-Hall Book Distribution Center, Route 59 at Brook Hill Drive, West Nyack, New York 10995. **OR. . .VISIT YOUR LOCAL BOOKSTORE**

*UNIX is a trademark of AT&T Bell Laboratories

CIRCLE 83 ON READER SERVICE CARD

```
PPL listing for best-curve fit

PROGRAM BEST_FIT

Version 1.0   June 10, 1984
LAST UPDATE :  06-15-84        12:22:23

INPUT Filename                -- Get name of data source
OPEN Filename,#1,INPUT
READ #1,Number_of_data        -- Read the amount of data
  INITIALIZE   None           -- Loop to read the (X,Y) data pairs
  LOOP
  BEGIN   For counter = 1 to Num_of_data
     READ #1,X(counter),Y(counter)
  END LOOP
  TERMINATE   CLOSE #1 -- close buffer

Set Num_Model = 4   -- This sets number of models

[DO REGRESSION]

Print sorted results

END PROGRAM
----------------------------------------------------
PROCEDURE [DO REGRESSION]

  INITIALIZE None
  LOOP   Outer
  BEGIN   For model = 1 to Num_Model
       INITIALIZE   Set regression summation to zero
       LOOP   Summmations
       BEGIN   For count = 1 to Num_of_data
          [Transform]
          Update summations using X and Y
       END LOOP   Summations
       TERMINATE   None
     Calculate Slope, Intercept and correlation coefficient
     -- Perform insertion sort using the values of the
     -- correlation coefficient
        [SORT]
  END LOOP   Outer
  TERMINATE   None

END

----------------------------------------------------
PROCEDURE [Transform]

  CASE   model

       WHEN   1 => X = X(count); Y = Y(count)
       WHEN   2 => X = Log(X(count)); Y = Y(count)
       WHEN   3 => X = X(count); Y = Log(Y(count))
       WHEN   4 => X = Log(X(count)); Y = Log(Y(count))

  END CASE

END

END PPL -- This must be included in every PPL listing using
        -- program NILE.PAS
```

Listing 5.

DESIGNERS DEBATE

UNIX on micros

By Ken Takara

UNIX has been around on minicomputers for a long time. This month, however, let's focus on the future of UNIX as a microcomputer operating system alternative.

Many factions in the microcomputer industry have stated that UNIX may never receive the market acceptance UNIX supporters claim it will primarily because of certain inherent problems with the UNIX operating system's overall organization and interface to the programmer and end user.

Let's turn now to a discussion of UNIX on micros and focus on certain arguments that deal with UNIX as a development environment.

The following people debate the UNIX issue in this month's column: John McNamee, a free-lance consultant; Jack Crenshaw, software engineer; Bob Peterson of the MUSUS SIG on the CompuServe data base; Mike Cohen; Morton Goldberg; Tim Smith; Bob Upshaw; Earle Robinson; Eli Willner; Joel Swank; and Dennis Hamilton.

Is it reasonable to run UNIX on a micro? It has a reputation as a rather large operating system, requiring vast amounts of memory and disk space.

McNamee: Define micro and you will get a better answer. I doubt a Commodore 64 will run it, and the IBM PC has a hard time. My view is that UNIX requires a CPU made by somebody other than Intel and a disk drive that is faster than the ST506 interface, 100-millisec seek drives you find in PCs.

Peterson: I have had occasion to look into how much machine UNIX needs. I came to the conclusion that 256K RAM, memory management/protection and 5MB of hard disk space are the absolute minimums for UNIX.

Certain clone systems—for example, IDRIS—require less disk space and no memory management. This places UNIX in the medium to upper ranges of micros at a usual cost of $5,000 through $7,000 for the basic machine. This would be for a single-user machine.

I tend to believe that UNIX is a better fit with the architecture of the MC68000 and National 32000 processors. In the Intel world, an 80286 is probably the realistic minimum for decent performance.

Cohen: 256K RAM? Try doing anything useful on a 256K TRS-16! 512K is more like it. That and a 5MB hard disk would still be a *very* tight fit. A nice thing would be a streamlined, single-user UNIX subset.

McNamee: That's not really true. The PDP-11 is a mini, and it can only address 64K of code plus 64K of data, and you can't even play tricks to get more. There are no user-accessible segment registers. The definitions of super-micro, mini, super-mini, and mainframe seem to constantly change. When the PDP-11/70 came out, everybody said it was almost a mainframe. Now you can get an LSI-11/73 with about the same speed right on your desk.

I think Fortune tried to sell floppy-based UNIX. You will notice that their profits (or should I say, lack of them) may have had something to do with trying to sell systems too small to run UNIX. Tandy threatened to release floppy-based XENIX, but so far it has not shown up. I don't know anybody else doing it.

Goldberg: I'll stick my two cents in. I had VENIX/86 running on my PC-XT. I killed it because it wanted too much disk space. It would only allow PC-DOS 3MB of the 10MB on the hard disk. My PC-DOS applications outgrew 3MB and VENIX had to go.

VENIX/86 is a real UNIX, an AT&T licensed version 7 with some Berkeley enhancements such as the vi screen editor. Code developed under VENIX should run on most UNIX systems if it is recompiled on the system it is moved to. I got VENIX to develop code for what I thought was going to be the booming UNIX-on-PCs market. But the market didn't boom.

I earn more money from stuff I do on PC-DOS than I could with the VENIX stuff. So it was an economic decision that forced me to abandon VENIX. If I could have afforded a second hard disk, I would have kept them both.

Crenshaw: It seems to me everyone's missing a key point: UNIX is a time-sharing system. That means that one CPU is used to poll or otherwise respond to a usually large number of user terminals. It also means a very considerable overhead is required to allow that CPU to do all the bookkeeping necessary to support that.

That means context switching, access control, virtual memory overhead, priority control, accounting, mail and message handling—not to mention the ubiquitous news!—etc., etc. That doesn't leave much CPU left over for number crunching, as anyone who's ever used a 40-user VAX can testify.

UNIX is without a doubt the best OS ever written for multi-user, time-sharing systems, bar none. Now the question is, do such systems have any place on a micro? As a strong supporter of Purnelle's law—at least one CPU per user—I claim they do not. For some reason, whenever a new computer is announced (for example, the IBM PC AT) the first question asked is, will it run UNIX? The next one is, how many users will it support?

My question is, why do we want to crowd people onto a single CPU? Has silicon suddenly become a precious metal? To get back to single-user UNIX, isn't that OS-9?

McNamee: There is no reason why UNIX can't be run single user. The only difference between multi-user and single-user UNIX is an entry in */etc/ttys*. A process is a process is a process. It doesn't matter if they are all attached to the same terminal or to multiple terminals.

I think the reason everybody asks if it runs UNIX and how many users it supports is that networks aren't doing the job they should. Right now it costs less to have one machine serving multiple users than it does to network a bunch of PC's.

Peterson: UNIX is one of the few operating systems that will live comfortably on a local area network. A single-user UNIX machine in a LAN satisfies the more-than-one processor/user criterion while providing multitasking. OS-9 is not UNIX, nor is it a single-user UNIX.

There is a need in the real world to share data. At the current time, a multi-user machine is the best understood environment for data sharing. The LAN environment is finally getting some software packages that share across the network, but they are few.

At the present time, a 3 to 10-user machine is cheaper than 3 to 10 micros. This is because of the mechanical devices and packaging, not the silicon. When the economics of the world change so that a LAN environment with data sharing tools is less expensive than a multi-user machine, and such a system is available from a trusted vender, then LAN systems will become more prevalent than multi-user systems.

Hamilton: I believe that UNIX having been implemented in a time-sharing setting is irrelevant to a discussion of what UNIX is. There isn't much about the architecture that users see that commits UNIX one way or the other.

You build it with a time-sharing kernel when you've got a large facility that you want to share. When you have small facilities that you want to interconnect, you do it differently. But the user, apart from observed performance differences, should be able to get work done on either with the same knowledge and skills.

Now, who would want to use the world's most impossible system . . . ?

McNamee: A programmer with an IQ greater than his or her shoe size should have no problem. Users are best given menu shells or another such interface. Turning a dumb user over to one of the standard shells is cruel and unusual punishment.

UNIX was a system designed by programmers for other programmers. It does make it very easy to set a user up with a menu shell, but you need a programmer to do the initial setup. I think the same could be said about CP/M or MS-DOS. The CP/M prompt A> doesn't give you much more to go on than + or %.

Peterson: UNIX is an old system. Many times this means that there is a lot of software available, and for UNIX this is true. But in the UNIX world, this software isn't oriented to the business or personal user but to the academic or software developer.

The fact that UNIX was designed in the early 1970s also means that it was based on contemporary design principles and hardware. This implies that UNIX itself doesn't know about bit-map displays, for example. Some applications do, and some venders do ship bit-map-oriented user interfaces, but these are add-ons, not a part of the basic system.

The average user will be better served with TopView on an IBM PC AT or with some other system for which appropriate applications software is available.

Upshaw: The only way to address the severe shortage of good software is to make it possible to write a piece of software once and use it on a large number of different machines.

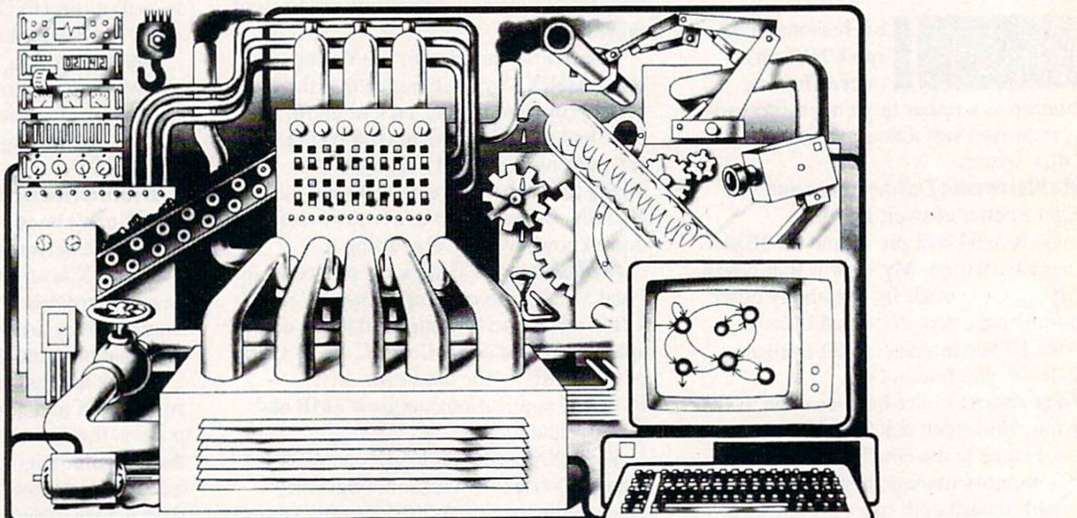
It's clear to even the most casual observer that UNIX is running on more different machines than any other operating system. It's only natural that UNIX migrate to micros.

Crenshaw: I think when many people talk about UNIX, they are really talking about the user interface. And that can be gotten in a lot of OS's. As a matter of fact, I think we should be sure we are all talking about the same aspects of the OS.

The part of UNIX that I think is least suited to micros is the file system and the terminal interface software. I love the directory structure and the shell.

McNamee: I am a programmer. I

Hard problems? **Csharp** can help!



Cut your development time with C source code for realtime data acquisition and control. The **Csharp Realtime Toolkit** includes: graphics, event handling, scheduling, and state systems. Processor, device, and operating system independent.

SYSTEMS GUILD

Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

CIRCLE 89 ON READER SERVICE CARD

know what I'm doing. I usually get things right. UNIX is for people just like me. It provides me with the tools to get my job done in the least amount of time with the least effort. It also allows me to easily create any user interface I desire.

I can set up a machine that secretaries can use without ever knowing about any UNIX commands. All they need to do is log in. The only people who lose with UNIX are the ones who need help (i.e., dumb end users) but don't buy a system with proper support (in the form of a programmer to set them up).

It is very important to remember that UNIX is not MS-DOS or CP/M. It is a multi-user operating system. I don't know of any good multi-user system that is so simple a total novice could make it go without help. Sure, it's easier to use MS-DOS, but MS-DOS does not solve the same problems as UNIX.

Smith: I like UNIX too, as a programmer's system. But I curse it many times, for example, when I forget some bizarre aspect of, say, shell syntax. Also, as a one-time UNIX system administrator on a big VAX, I know how bad it can be when you have a bunch of dumb and/or unfriendly users aboard.

We need something better. There's no reason why it can't be done. Maybe it's mice and icons for the end users and a good shell for the experts, both on the same system. But it's going to have to be a lot more efficient than any current UNIX.

Robinson: I can certainly agree that UNIX is better than CP/M, which is only a very primitive file loader. But to say that it is better than MS-DOS for a single user is something else. UNIX, due to its constant file accessing, would be unacceptably slow on most micros unless it were a CPU of the MC68000 family.

Otherwise, with the features now available under MS-DOS and the excellent public domain utilities—like directory programs, extended type-ahead buffer, NDOSEDIT, editors, squeeze and unsqueeze programs, shells, etc.—MS-DOS is the system of preference for most people other than those actually doing program development.

McNamee: Does *dir* really mean more that *ls*? I doubt it. Most of the places where I've seen a large number of dumb users doing computer work, they all had their instructions written down, and they just followed the steps. They really didn't know what they were doing, and in this kind of environment, it doesn't matter whether you type *delete*, *era*, or *rm* to erase a file.

Crenshaw: Count me in as one who has been less than overwhelmed by UNIX. Well, to be more accurate, the thing that turns me off most is not so much the system as the aura of mystery that seems to surround it. It's what I call the Fraternity Syndrome.

In brief, a young computer science student gets his first exposure to UNIX in school. He's totally blown away by it and totally in awe of the UNIX wizards. But after stumbling around for a few years, he learns some things and wakes up one morning to find that he's one of the wizards. He's held in awe by others and that feels pretty good.

Now do you think he's going to tell you all you need to know to use UNIX effectively? No way. You're going to have to pay your dues, same as he did.

I find that this attitude pervades everything about a typical UNIX installation, including the command names and the documentation. To me, it takes a lot of the fun out of things. It's like some kind of gigantic inside joke.

Willner: One more vote against UNIX. I don't like to fight my environment. I don't like to memorize archaic commands, no matter how much it makes me look like a wiz. I don't like to destroy all my files because of a misplaced keystroke.

Swank: UNIX is hard to learn, but once learned, it is very powerful. I can do in one line in UNIX what would take me 200 cards in IBM-360 JCL.

Peterson: Willner pointed out a couple of the reasons I don't like UNIX. The command names don't relate to the function of the command and I, too, don't like an environment where dangerous operations don't prompt for verification.

However, my dislike goes deeper. UNIX doesn't have a consistent method of record locking, and the file system is excessively fragile. The OS doesn't take advantage of the fact that almost all terminals are now CRTs, not TTYs. Not to mention that UNIX doesn't understand bit-mapped displays.

One of the great myths about UNIX is that there is a single thing named UNIX. This just isn't so. User groups are just now defining what they consider the core of UNIX, including OS calls and library utilities. In addition to the various versions delivered by AT&T and Berkeley, each processor's and/or vender's implementation differs in subtle ways.

I like the tools philosophy embodied in UNIX and the concept of pipes. However, the idea of pipes deserves to be extended. It hasn't really changed since UNIX was first written up in SIGPLAN notices.

Upshaw: The UNIX user interface allows one to easily go from one machine to another. Sure VisiCalc will port fairly easily from one CP/M machine to another (say, an Apple to a Xerox 820), but it's a rewrite to port to an IBM PC, not to mention a VAX or a Cray (yes, there is a spreadsheet on the Cray). And, of course, the UNIX/C environment is conducive to portability.

McNamee: I like UNIX as a programmer's environment and would hate to see it turned into a pure end-user system. I think it is quite possible to build a reasonable interface on top of existing UNIX. I don't think you can build a programmer friendly interface on top of a Mice/Icon system.

In its intended environment, the Mice/Icon interface works well. It just doesn't do anything good for me as a programmer, and in fact, it holds me back. I want to bring up again my point that it isn't fair to compare the user friendly aspects of UNIX to existing single-user OS's.

Compare it to other time-sharing systems. Multi-user systems such as UNIX were created with goals very different from those of the Mac. UNIX is good for what it was designed, and the Mac is good for what it was designed. UNIX is not the OS for everybody or even for the majority of the people.

Upshaw: When I am interviewing candidates for a job, more often than not they have used UNIX. UNIX is very popular in the universities, and with all the major venders jumping on the UNIX bandwagon, its popularity is likely to increase.

When I hire someone, I don't want to train my employee to use five different systems—I've got more important things to do. If he or she already knows the system that runs all five of my machines, I'll save a considerable amount of time and money.

Smith: Try to get hold of a copy of the November issue of *UNIX Review*. There's an interview with Don Norman, a professor of psychology at Univ. of California at San Diego. In it he explains why he does and doesn't like UNIX. He's a sharp guy and has become something of an expert in the design of systems for good human interaction.

Periodically, I will be holding argument clinics over some subject on the CLM SIG on CompuServe. If you don't happen to have a CompuServe account, you can still join in. I will try to announce the future subjects in advance on the *COMPUTER LANGUAGE* Bulletin Board Service, so send your thoughts through the mail or via the BBS. ■

THE MAGAZINE FOR THE FULL-OUT BRAINO.

COMPUTER
LANGUAGE
PREMIER ISSUE
A LIGHTNING FAST C SCREEN EDITOR
CBASIC VS. MBASIC
MACRO LIBRARY FOR CROSS ASSEMBLY IN PASCAL
NATURAL LANGUAGE PROCESSING

Computer Language is written for people who can program in two or *more* computer languages.

Let's face it, that leaves out most people. Programming is a rigorous, intellectual discipline and Computer Language magazine is the first and only publication dedicated exclusively to this field.

Your source for the latest technical skills and methods used by software specialists.

We cover the major developments in the software design field, from theory to implementation. Computer Language focuses on the most important and useful language design information available in the fast-moving microcomputer industry.

Written for the person who takes computing seriously.

We're talking about you — the experienced software author, programmer, or engineer who routinely programs in two or more high-level languages.

A person who understands the creative nature of programming and appreciates the beauty of efficient code in action.

COMPUTER LANGUAGE will constantly challenge your abilities.

The foremost industry experts will discuss: · Algorithmic Approaches to Problem Solving · Language Portability Features · Compiler Designs · Utilities · Artificial Intelligence · Editors · New Language Syntax · Telecommunications · Language Selection Criteria · Marketing Your Own Software · Critical Software & Hardware Reviews

Send to:

COMPUTER
LANGUAGE
2443 Fillmore Street Suite #346 San Francisco, CA 94115

YES! Start my charter subscription to Computer Language. My 1 year charter subscription is just $24.95, a 33% savings under the single copy price.

☐ $24.95 Payment Enclosed ☐ Bill me.

Name

Address

City State Zip C

COMPUTERVISIONS

P.J. Plauger reflects on the history of C

By Craig LaGrow

He calls himself a programmer at heart. But as an accomplished science fiction writer, president of a successful software company, technical book author, musician, and runner, P.J. Plauger is certainly a man of many talents.

His friends and work associates know him as Bill—a name he was affectionately given by his older sister three days after his birth. But in any published, professional work, he prefers to use the initials of his real name, Phillip James.

Plauger's first contact with Bell Laboratories was as a undergraduate at Princeton Univ. in the early 1960s. He and several of his fellow students were fortunate enough to shuttle back and forth between Bell Labs and school and get involved in temporary summer projects.

"In those days it was heaven," says Plauger. "It was the place to go because just about anything you could do that would advance the state of communications in the next five years was considered fair game. And it was the place to be if you were interested in being at the cutting edge of programming."

"In the 1960s, there was still this tremendous intellectual freedom in the very undefined world of computer programming," he remembers. "Maybe a place like IBM Yorktown Heights, RCA, GTE, or XEROX would come close, but Bell Labs was really just a wonderful place to be for freedom and access to resources."

Plauger went on after Princeton to earn a Ph.D. in nuclear physics from Michigan State Univ. There he spent half his time, with a fellow graduate student, designing and writing a virtual memory operating system. He confesses that he "invented" demand paging, not knowing that the rest of the world had already done so.

"I've worked and earned my living as a programmer since 1963, and I say that very proudly," says Plauger. "I consider myself a programmer rather than a manager or a computer scientist. Those are things that I've done, but I've earned my living as a computer programmer."

Plauger started working for Bell Labs upon receiving his doctorate in 1969, first at Holmdel, N.J., then later at their Murray Hill, N.J., headquarters. He spent his five and one-half years at Bell Labs as a member of technical staff, which is the standard staff level for people doing research. His department was chartered with getting computers into telephone central offices and digitizing things so that data processing technology could be better brought to bear.

But he was also surrounded by people like Dennis Ritchie, Ken Thompson, and Brian Kernighan, who were working on the beginnings of what was to become UNIX and C. He naturally soaked up as much of that technology as he could, fascinated by anything that could improve programmer productivity.

While Plauger was working at Bell Labs, he designed small, in-house languages, built a few compilers, and began to work with rudimentary UNIX and program development tools.

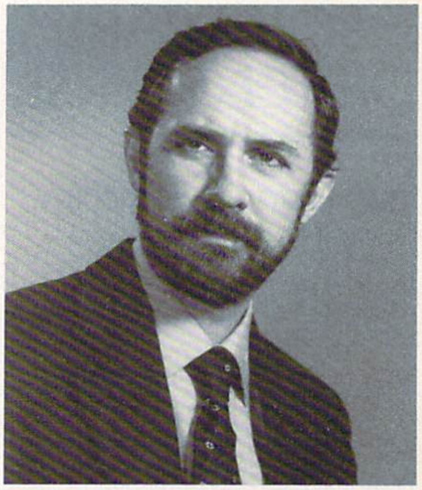
"In many ways, the success of UNIX was little more than a matter of Thompson and Ritchie setting up fences, if you will, to herd people in the right direction," remembers Plauger. "They just made it easier for people to write reusable software with good properties. If there has ever been a central committee on how to homogenize UNIX, it has only existed the last couple of years—and a lot of the payoff has been lost in the growth and size of UNIX from attempting to homogenize it."

According to Plauger, there is no question that Ritchie invented the C language and made it what it is today.

"It was his baby, and that's one of its great strengths; its integrity stems from the fact that there was one good designer who was really responsible," says Plauger. "All of us to this day who are involved with it deferred to him."

Plauger was one of the first real C programmers. Working on an old PDP-11/20—the very first PDP-11 UNIX system—he recalls tripping over many of the bugs when things like structures first went into the language.

An interesting footnote to the C language development going on at the time was that Plauger was simultaneously developing a language he hoped would rival C. He then completed a language implementation that today would resemble something like PL/M for the PDP-11.



"One of the smarter things I did in my career was realize after a period of time that the C language was superior to mine," he says. "I actually had the decency to stand up and tell a lot of people that even though I'd put a lot of work into this language, C was better and they should use it instead. I became a convert in the early days."

Thompson had been experimenting with a language called B, a precursor to C that was, according to Plauger, a "cute derivative" of another experimental language called BCPL. "The trouble with it on a PDP-11, though, was that it assumed everything was a 16-bit integer," Plauger recalls.

The B language could manipulate pointers like 16-bit integers but, because of its weakness with byte resolution addressing of the PDP-11, a programmer had to know to count by twos to step through an array. Because of this, data typing went into the language just to help the compiler know how to manipulate pointers to different types of data.

The second problem with B was that it wasn't a true compiler but rather an interpreter.

"C in a nutshell is B with typing to get

C

SOFTWARE DEVELOPERS!

V - FILE
THE VIRTUAL MEMORY FILE MANAGER

Let V-FILE save precious development time & cost as you create efficient applications with the power of VIRTUAL MEMORY.

DON'T RE-INVENT THE WHEEL

Why spend weeks or months coding and debugging file and memory management systems when you can order V-FILE today. V-FILE is a library that you can link with your code to provide sophisticated virtual file and memory management — allowing you to concentrate on developing your application.

VIRTUAL DATA OBJECTS SUPPORTED!

Data is referenced by using VIRTUAL MEMORY DATA HANDLES. Your code doesn't need to know whether the data is actually on disk or in RAM. Swapping between disk and RAM and updating files on disk is handled automatically and transparently! Complex VIRTUAL DATA STRUCTURES can be created by linking with data handles instead of pointers.

CHECK THESE FEATURES!

- Multiple, independent swap buffers
- Multiple files per swap buffer
- Highly efficient swap algorithm
- Automatic file updating
- Data prefetching supported
- Data may be locked in memory
- Memory buffers may be flushed
- Makes full use of extended memory on IBM PC/AT
- SOURCE CODE AVAILABLE
- NO ROYALTIES REQUIRED

Supports Dos 2.00+ with Lattice & Microsoft C compilers
Supports Microsoft windows

$299

V-FILE TM

Contact:
MindBank, Inc.
4620 Henry Street
Pittsburgh, PA 15213
412/683-9800

VISA/MASTER CARD ACCEPTED

CIRCLE 63 ON READER SERVICE CARD

the pointer arithmetic right and true compilation to get better efficiency," says Plauger.

Ironically, however, all the development work on UNIX and C at the time was being done in total opposition to the administrative philosophy of Bell Labs. The fact that the project survived, according to Plauger, was mostly due to a policy of "salutary neglect" on the part of management.

"There were no projects, no specs, no requirements; it was almost all done on speculation—all of it," says Plauger. "Over about a two-and-one-half-year period, one by one most of the people in our area at Bell Labs elected to do their work on UNIX, but nobody told them to," he says.

The GE 635 computer, which was one of the large systems that Bell researchers were supposed to be using in their work, actually turned into a remote printer for the UNIX machine as far as this group was concerned.

"Nobody made them do that. Nobody made them write up the utilities which eventually got swept up into the UNIX package. Nobody enforced a uniform style," says Plauger. "As a consequence, we can all now piss and moan about the irregularities that are there. But considering there was no enforcement, it's remarkably regular. The miracle is that UNIX is as good as it is, not that it couldn't be better."

The early prototypes to UNIX and C also were being used for many interesting projects in-house at the time. Plauger often reminds people that Thompson started UNIX to support his own "little solar system exploration thing" so it would be easier to develop the software for it.

"There were half a dozen people hanging around the UNIX room because each had his own little pet project going, and UNIX was the best leverage to get that accomplished," says Plauger. "Not a single one of them was chartered by management in order to improve productivity or in order to pursue that particular goal."

"It was very easy to get caught up in all this," he recalls. "Of course having Brian Kernighan next door to shoot the bull with and speculate about what we could do with all these things was great."

As Kernighan and Plauger became closer friends and work associates, they ended up writing two books on programming style called *The Elements of Programming Style*,[1] *Software Tools*,[2] and *Software Tools in Pascal*,[3] which have become text book standards in many computer science curricula across the country.

"I think Brian had a folder of examples that showed how bad the training for programmers was at the time," he says. "Finally, we'd get to a 'there ought to be a law' stage."

"What we did with *The Elements of Programming Style* was to write the manuscript fairly quickly and circulate it among our friends," he recalls. "We'd tend to go on a kind of feeding frenzy, tearing up programming examples from other books. It took us a couple of iterations to get to the correct style that I think made the book what it is.

"Brian is a very solid writer, very direct. And I've tried to emulate Issac Asimov's science-fact writing," says Plauger. "Between the two of us, Brian and I managed to keep the worst offenses of our programmer arrogance and love for big words out of the document. It managed to be something better than either one of us could have done on our own."

Plauger remembers Ritchie being a bit more distant to him both professionally and socially at first. "He lived in his own world, which was populated by the likes of Ken Thompson and Doug McIlroy," says Plauger.

"It was like, wow, this guy comes in at 1:30 or 2:00 p.m. and immediately goes to lunch! But nobody seemed to mind because he'd work till midnight with these people, and they'd all get tremendous amounts of brilliant work done," he says.

Plauger admits that he'll always have a tremendous respect and admiration for Ritchie. "He's a marvelous writer and an excellent designer with what amounts to a slightly enhanced version of Appendix A in Ritchie's and Kernighan's famous book, *The C Programming Language*.[4] Also of importance was the /usr/group effort on standardizing libraries.

Serving as the secretary of the committee since its inception over a year ago, Plauger was also appointed chairman of the Library TG—one of the three technical subcommittees working on pieces of the soon-to-be published ANSI X3J11 C standard.

"I think it's amusing that I ended up as chairman of the Library TG because some have said that's like putting the fox in charge of the hen house," jokes Plauger. "I gravitated toward that because of the tremendous technology that my own compiler company [Whitesmiths Ltd.] has developed for making C work well—not just in UNIX but everywhere."

"We all agreed to adopt the /usr/group standard as the base document for the library standards effort. Essentially what I did was show the committee how to modify that set of functions to get the UNIX-isms out, so that they could be implemented in a variety of environments.

"We also have put forward and adopted a number of additional functions which you need outside of the UNIX environment. The net result is that what ANSI now has is functionally very compatible with UNIX V7 and System III, which /usr/group has standardized on, but is strongly enhanced semantically to make a more widely usable standard," says Plauger.

Participating in the ANSI committee meetings have been representatives from over two dozen C compiler companies. Consequently, the interests of many companies comes to bear in each of the standards meetings.

According to Plauger, compiler manufacturing companies have a strong influence on the outcome of the standardization. "Other groups, such as educators and independent consumers, are often outvoted because of this makeup," he claims.

"Obviously one of the reasons that we're having all these standardization meetings is that we have these different dialects that have evolved in different little communities. What we're trying to do is to bring them together."

Plauger feels strongly that the main purpose of the ANSI standards effort is to create a marketplace for the C language to grow into.

"If there is no marketplace, there's no way you can make money," says Plauger. "You can bury your gold in your back yard and be safe, but that's not a market. Or at the other extreme, if there are no rules for safe trade, there'll always be pirates waiting out there every time you send a ship out to do business. It'll get boarded and everything gets ripped off. That's not a marketplace either."

But today, after a year and one-half of meetings, the committee is preparing to finalize a document that will formalize the rules and definitions that future C compiler writers will follow when designing different implementations of C for different hardware and operating system environments.

"In these last few meetings, we're really just getting down to the sorts of things that Talmudic scholars would sit around arguing about on a Saturday," says Plauger. "We're really just quibbling over whether certain nice extensions should get stuck into the language before it's frozen and what the subtle interpretation is of certain types of coercions. Things like that."

Yet Plauger is also willing to admit that he thinks the C language will soon begin to decline in popularity within the programming community as a whole.

"My feeling is that C is just short of peaking in popularity—that is, it has almost peaked and will start declining soon," says Plauger. "C has some maintenance problems with large programs. It's a little too lax in its declarations and its type checking, and it doesn't have all the facilities you need, such as for packaging."

"You see the amount of stuff that's been written in C, from commercial applications to you name it," says Plauger. "FORTRAN ranks in popularity right up there with C, but C is unquestionably the better language right now. Its major limitation is that when you get a programming project somewhere between 50 and 200 and 1,000 lines of code, you start losing control."

"C just doesn't have all of the things that you need for modularization, strong checking, and so on. It's evolving in that direction, but it's got its weaknesses. Today certain dialects of Pascal are becoming fairly powerful too. The trouble with Pascal is that in order to meet those requirements it has to be extended, and nobody seems to have agreed on which extensions yet.

"Ada is still out of the starting gate. If I had to do a large project commercially, I would do it in structured COBOL. If I felt the demand for performance were too great, I might switch to PL/I but probably to C," says Plauger.

Bell Labs is preparing to introduce an enhanced version of C, called C++, which, according to Plauger, "has a lot of bright ideas tacked on. But it's not compelling to me that's the best thing to do to C yet. Certainly a lot of good ideas that are missing in C can be found in Ada. Ada is full of a lot of things, it's just too big a language. I think right now programming in C, with tighter discipline, is what we're going to be relying on to carry us through the next few years," says Plauger.

Plauger's five-year stint with Bell Labs did not end on the best of terms. In 1974, the woman who he was married to at that time filed charges of sex descrimination against Bell Labs, and Plauger decided to support her in her suit against the company because he felt her claims were justified.

"I think that because I was so outspoken in encouraging her to oppose the company made it hard for people to want to keep me on at Bell Labs," he says. "Let's face it, I didn't have the personality to understand how to work well in an organization of that size. I had a chance to leave under my own steam before I got run out of town on a rail, so I took it."

Plauger left Bell Labs in 1975 to work for Yourdon Inc., a seminar and book publishing company based in New York, N.Y. Ed Yourdon, founder of the company, had created a strong reputation worldwide for being an author of important books on programmer productivity.

"I'd say I went to Yourdon with a head full of ideas about programming tools," he remembers. "When the opportunity came up to actually write a C compiler, I was quite confident that I could do it even though I hadn't written a whole C compiler before."

But Plauger soon left Yourdon to start his own company on the kitchen table of his apartment on New York's Manhattan

YOUR CODE MAY BE WASTING ITS TIME!
THE PROFILER™ CAN HELP . . .

- Statistical Execution Profiler
- Works with any language
- Completely configurable
- Up to 16 partitions in RAM/ROM
- Time critical code optimization
- Abnormal code behavior tracking
- Graphic presentation of results
- Easy to use menu interface

THE PROFILER is a software package which gives you, the programmer, a powerful tool for locating time consuming functions in your code and allows you to performance tune your program. With the **THE PROFILER** you can determine where to optimize your code for maximum benefit, then measure the results of your efforts.

Using **THE PROFILER**, you can answer questions like:
Where is my program spending its time?
Why is my program so slow? What is it doing?
Is my progam I/O bound? CPU bound? Are data buffers large enough?
How much improvement did my changes make?

THE PROFILER is completely software based and consists of a system resident driver and a monitor program. The memory partitions can range from 1 byte to 1 megabyte in size and can be anywhere in the address space.

NO ADDITIONAL HARDWARE IS REQUIRED!

Requires an IBM PC or compatible system with a minimum 64k and one drive.

THE PROFILER is available for $125.00 from DWB Associates or ask your software dealer. To order or for more information, call or write DWB Associates. VISA/MC accepted. Dealers welcome.

dwb
Associates
P.O. Box 5777
Beaverton, Oregon 97006
(503) 629-9645

IBM is a trademark of IBM Corp. MSDOS is a trademark of Microsoft Corp.
THE PROFILER is a trademark of DWB Associates.

CIRCLE 20 ON READER SERVICE CARD

island. He called the company Whitesmiths because it had a certain Old World sound to it. "We wanted to call to mind the old craft guilds almost back in the Middle Ages, certainly the early Renaissance," says Plauger.

Even the company logo—which shows a hand holding a ink quill pen in front of a collection of different lettering styles spanning thousands of years—can be seen both as an ancient Roman seal and a cathode ray tube.

"I have learned to think of programming not so much as an art but as a craft, meaning a more pragmatic art form," he says. "Certainly you don't write a program and just have everyone stand around and admire it."

Since that time, Plauger's company has written C compilers that stretch from the small, 8080-based microcomputers to the IBM 370. Most of their customers, he claims, depend on the fact that Whitesmiths is able to bridge such a broad range of systems and environments with a uniform language.

"Let's face it, I started this company because I like writing code and showing the world that you could really produce high-quality code that is marketable. I think we've done that," says Plauger. "Now I'm learning how to be a businessman.

"When you own your own company it just tends to eat you up," he says. "I do pride myself on being a writer, but I've had little time for that lately. And if you ask me what I do for a living, I'd say I'm a programmer."

Not only is Plauger the head of his own successful software company, but he's also a talented science fiction author. He won the John W. Campbell Award for best new science fiction writer of the year in 1975. His story "Child of All Ages,"[5] originally published in *Analog*, has been widely anthologized and has been optioned for television. His novel, *Fighting Madness*,[6] and several other short stories have been published by *Analog* and other magazines.

Plauger emulates the prose style of C.S. Forrester in his fiction, just as he models his technical writing after Asimov. He prefers to write in a clean and straightforward way, without forcing his readers to "bump their shins on the big words or funky grammar," as he puts it. "When you write science fiction, you can't snow people with fancy terminology, you have to sell a story about people."

Plauger now lives in a beautiful Early Victorian house in Concord, Mass., and spends most of his life with his family shuttling back and forth between this home and another lakeside home in New Hampshire. Above all, he says he enjoys watching his five-year-old son, Geoffrey, growing up.

Aside from programming and running a company, Plauger practices piano daily and manages to get in a jog every day as well. "I'm a runner just for the pleasure of it, enough to keep my heart ticking well and let me eat extra dessert."

As a multitalented individual, with interests spanning from the artistic to the pragmatic, Plauger certainly qualifies as one of the more fascinating Renaissance people in our computing industry today.

References

1. Kernighan, Brian W. and Plauger, P.J. *The Elements of Programming Style*. McGraw-Hill Book Co., New York, N.Y., 1974.
2. Kernighan, Brian W. and Plauger, P.J. *Software Tools*. Addison-Wesley Inc., Reading, Mass., 1976.
3. Kernighan, Brian W. and Plauger, P.J. *Software Tools in Pascal*. Addison-Wesley Inc., Reading, Mass., 1981.
4. Kernighan, Brian W. and Ritchie, Dennis M. *The C Programming Language*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1978.
5. Plauger, P.J. "Child of All Ages," *Analog*, (Nov. 1973).
6. Plauger, P.J. "Fighting Madness," *Analog Annual* (1976).

Craig LaGrow is the editor of COMPUTER LANGUAGE.

SMALL C FOR IBM-PC

Small-C Compiler Version 2.1 for PC-DOS/MS-DOS
Source Code included for Compiler & Library
New 8086 optimizations
Rich I/O & Standard Library

$40

CBUG SOURCE LEVEL DEBUGGER FOR SMALL C

Break, Trace, and Change variables all on the source level
Source code included

$40

Datalight
11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

ASM or MASM is required with compiler.
Include disk size (160k/320k), and DOS version with order.
VISA & MasterCard accepted. Include card no. & expiration date.
Washington state residents include 7.9% sales tax.
IBM-PC & PC-DOS are trademarks of International Business Machines
MS-DOS is a trademark of Microsoft Corporation.

CIRCLE 19 ON READER SERVICE CARD

Only $95 with FULL SOURCE CODE!

Q/C

"... an incredible learning tool." *Byte*

For only $95, Q/C is a ready-to-use C compiler for CP/M with complete source code. Here's what *BYTE* (May 1984) said: "Q/C ... has a portable library and produces good code quality. If you want to learn compiler construction techniques or modify the standard language, Q/C is the obvious choice."

- Source code for compiler and over 75 library functions.
- Strong support for assembly language and ROMs.
- No license fees for object code.
- Z80 version takes advantage of Z80 instructions.
- Q/C is standard. Good portability to UNIX.

Q/C has casts, typedef, sizeof, structure initialization, and function typing. It is compatible with UNIX Version 7 C, but doesn't support long integers, float, parameterized #defines, or bit fields. Call about our new products: Q/C profiler, Z80 code optimizer, and Z80 assembler and virtual linker, all with full source code!

THE CODE WORKS
5266 Hollister, Suite 224
Santa Barbara, CA 93111
(805) 683-1585

Q/C, CP/M, Z80, and UNIX are trademarks of Quality Computer Systems, Digital Research, Zilog, Inc., and Bell Laboratories respectively.

CIRCLE 42 ON READER SERVICE CARD

LOMAS DATA PRODUCTS INVITES YOU TO:

SHARE THE THUNDER.

The S100-PC-TM offers the following standard features:

- ☐ High performance THUNDER186 8Mhz 80186 processor
- ☐ 512K bytes of RAM (expandable to 1Mbyte)
- ☐ 4 serial ports to support up to four users
- ☐ 3 Centronics compatible parallel ports
- ☐ Concurrent DOS operating system allows execution of both CP/M-86 and MS-DOS (PC-DOS) programs
- ☐ 5¼″ IBM-PC compatible floppy drive
- ☐ 40 Mbyte high performance Winchester drive
- ☐ Attractive 10 slot desktop enclosure

In addition, a number of options are available including: larger Winchester drives, more user ports, 80286 processor, graphics support and additional operating systems (MS-DOS and CP/M-86).

S100 BUS boards products & support for the system integrator . . .

All of LDP boards are fully tested to exacting standards and carry a one year warranty. We specialize in 16-bit products & support the major operating systems for 16-bit processors: CP/M-86*, CONCURRENT CP/M-86*, and MS-DOS (PC-DOS).

■ THUNDER186 — THE ONLY COMPLETE S100 BUS, 16 BIT SINGLE BOARD COMPUTER. AVAILABLE TODAY.
Concurrent CP/M-86, which in addition to running CP/M-86 programs, runs MS-DOS programs. Comes complete, ready to plug into an enclosure and run. 256K bytes of RAM only . . . . **PRICE $1595.00**

■ LIGHTNING ONE*8086/8088 CPU**
8086 or 8088, with 8087 and 8089 coprocessors. Up to 10 MHZ operation . . . . **PRICES start at $425.00**

■ HAZITALL SYSTEM SUPPORT BOARD
2 serial, 2 parallel ports, battery protected clock calendar. Hard disk controller host interface . . . . . . **PRICE $325.00**

■ LDP 128/256K DYNAMIC RAM
Advanced dynamic RAM with LSI controller for failsafe operation, parity **PRICE** 128K—**$495.00**, 256K—**$795.00**

■ RAM67 HIGH PERFORMANCE STATIC RAM
High speed (100ns) low power CMOS static RAM. 128K bytes, extended addressing . . . . **PRICE $995.00**

■ LDP72 FLOPPY DISK CONTROLLER
Single/double density, single/double sided disks, both 8″ and 5¼″ inch drives simultaneously . . . . . **PRICE $275.00**

■ LIGHTNING 286—80286 CPU BOARD
Offers 4 times the performance of a 5MHZ 8086 CPU while maintaining software compatibility . . . . . **PRICE $1395.00**

■ OCTAPORT 8 PORT SERIAL BOARD
0 to 19200 baud operation real time clock interrupt. Ideal for multi-user systems such as MP/M-86* **PRICE $395.00**

S100-PC-TM is a trademark of Lomas Data Products, Inc.
**CP/M-86, MP/M-86 and CONCURRENT CP/M-86 are trademarks of Digital Research. **MS-DOS is trademark of Microsoft.*
****Lightning One is trademark of Lomas Data Products, Inc.*

S100-PC-TM: The LDP Multi-user S100 Bus System offers high performance at a "low" price . . . plus, "our" system is expandable and upgradeable!

PRICE
An unbelievable **$6995.00**

Call today!

LDP

LOMAS DATA PRODUCTS, INC.
66 Hopkinton road, Westboro, MA 01581
Tel: (617) 366-6434 ☐ Telex: 4996272

Dealer inquiries invited.

For orders outside the U.S., contact our exclusive dealers: ☐ **Australia** — LAMRON PTY. LTD., (02) 85-6228
☐ **Malaysia** — EXA COMPUTER (M) SENDIRIAN BERHAD, 795284

CIRCLE 54 ON READER SERVICE CARD

The Standardization of C

By Jim Brodie, chairman of C Programming Language Standards Committee

The C programming language has become widely accepted as one of the leading choices for a variety of applications, including systems programming (such as operating systems and compilers), industrial real-time control applications (such as telephone switching systems and process control), and even general applications (such as text editors, data base systems, and spreadsheets).

This widespread interest and use in many environments has prompted the development of a standard to enhance the portability of C programs by precisely specifying the language, library, and environment requirements of C.

In this article I will discuss a little of the history behind this standardization effort, the goals and working principles of the standards committee, and how the language currently being defined by the standard committee is different from the existing de facto standard for C. I will not address the standard activities related to the library or environment.

Why a new standard?

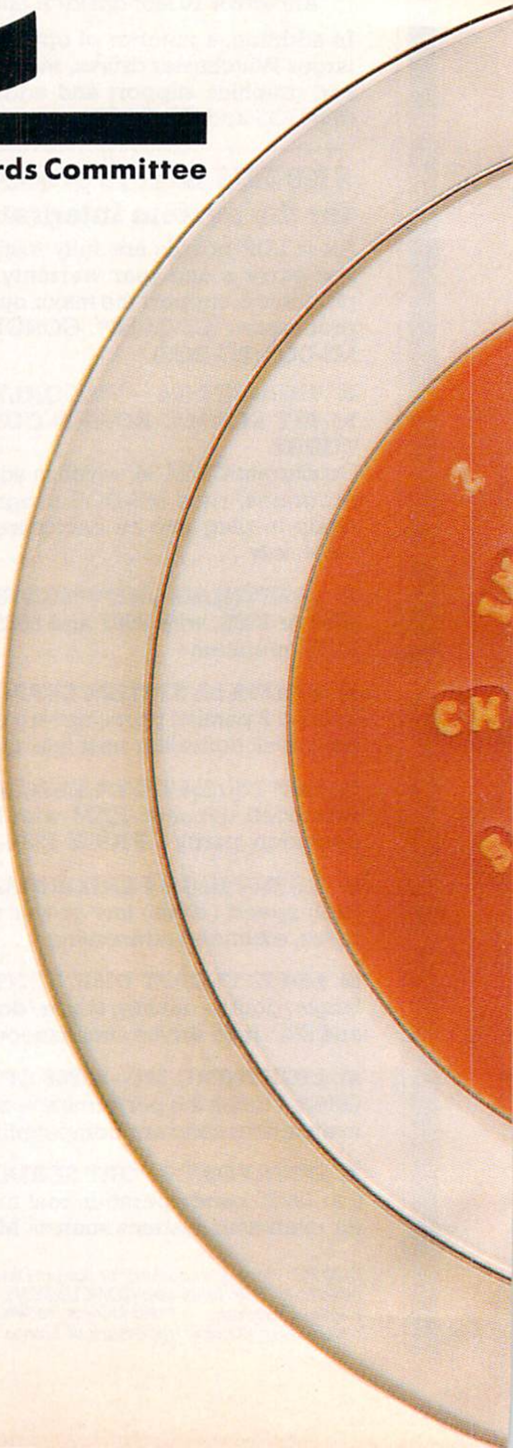
The C language has a de facto standard definition in *The C Programming Language* by Brian Kernighan and Dennis Ritchie (affectionately known as K&R in C programming circles). The language as described in K&R has been the starting point for almost every C translator (compiler or interpreter) in use.

Despite the high quality of this book, considerable divergence has occurred in the C translator implementations where language and library features or properties were ambiguously or imprecisely defined (or simply not defined at all). In addition, the language has grown in various directions as it has matured, and user requirements and language weaknesses have been identified. Unfortunately these extensions to the C language have varied significantly across C translator implementations.

Because of the variations in the different dialects of C, the portability of C programs—one of the often noted strengths of C—has suffered.

It became clear a need existed to define a standard that would establish a precise definition of what C translators should accept and what C programmers could count on when trying to write portable programs. This led to the formation of the C Programming Language Standards Committee (labeled X3J11) under the auspices of the American National Standards Institute (ANSI).

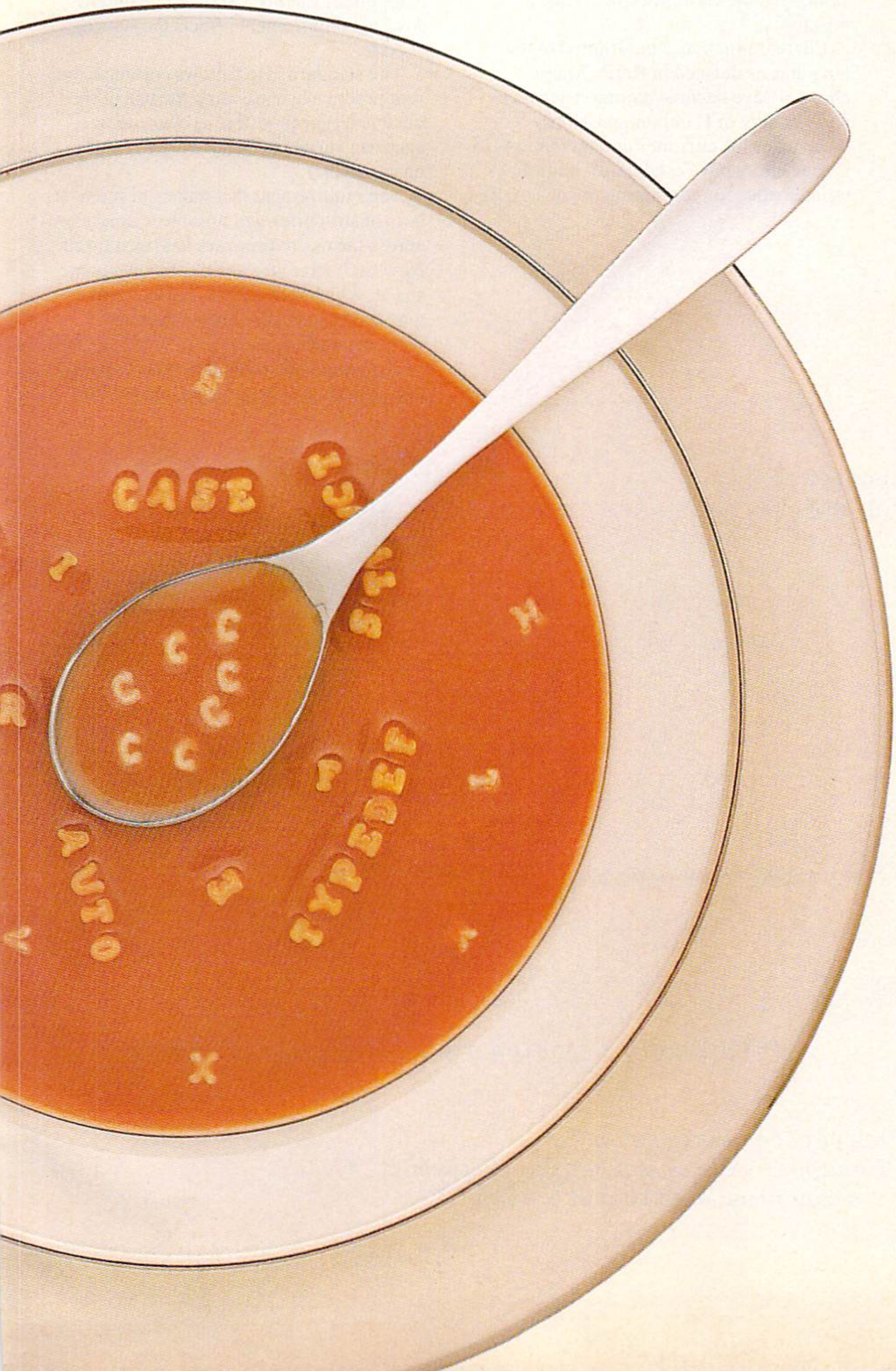
The standards committee, which is made up of volunteer representatives from a wide cross section of industry, aca-



demia, and C users, was formed in June 1983. The committee meets four times a year in one week sessions to formulate the new standard that will be presented to the C community for comment and, if satisfactory, approval.

The primary goal for the standards committee is to codify existing practice. It is not in the committee's charter to develop a significantly different language with many new features. Much of the work of the committee centers around determining what is existing practice and trying to resolve the differences when existing practice is inconsistent or contradictory.

The committee feels strongly that the resulting standard should enhance the portability of C programs. The committee uses the rule of "the fighting chance"—it is striving to develop a standard that does not force a C programmer to write portable programs (the low-level, machine-dependent features of C are part of its strength) but at least gives the programmer a fighting chance of writing a portable program if good programming practices are followed.



Maintaining the Spirit of C is also high on the list of the committee's goals. Some philosophical principles underlie the original design of the C language. The committee is striving to ensure that the C language it defines stays very close to the principles that made C successful in the first place. Some of these principles could be summarized in phrases like "Trust the programmer and allow the programmer to do what needs to be done," "Keep the language small and simple, and "Make it fast, even if it is not guaranteed to be right."

This last principle may need a little explanation. Many operations are defined to be "how the target machine's hardware does it" rather than by some general rule. For example, the size of the basic integer variable type, *int*, is defined to be the natural (read fastest) integer size for the machine. Another example is that C does not define whether there is zero fill or sign fill when an object is shifted to the right. It is allowed to be whatever is fastest and easiest on the particular target machine.

The astute reader may have noticed that the goals of allowing the writing of portable programs and permitting machine-defined features are frequently diametrically opposed. A significant part of the committee's work is determining, on a case-by-case basis, how the language can be defined so that each of the goals is reached in a reasonable way.

The committee is also working to define a standard that will break as little currently working code as is reasonably possible. It has generally been accepted that every translator will need to be modified to meet the requirements of the emerging standard. However, it is the committee's hope that only a small subset rewritten to be accepted by a standard conforming translator.

Other goals, including a desire for semantic precision, internal consistency, simplicity, and reliability, are also considered when trying to evaluate alternative approaches and solutions before the committee. These goals often come into play when considering possible extensions to the C language.

Changes to K&R

The language portion of the standard is being derived from *The C Reference Manual* by Dennis Ritchie, an early version of which was published as Appendix A of

The C Programming Language. In the remainder of this article I will outline some of the differences between the current draft of the standard and the language as defined in K&R.

It should be noted that the draft is not final and that the committee may change its position on any of the features described here. It is not possible to give a final definitive list of language changes and features since the committee remains undecided on several fundamental and many peripheral issues.

Despite this caveat, the features described here will probably be in the draft proposed standard submitted for approval unless a significant amount of C community feedback is received indicating that the committee has made a mistake.

Current practice is not limited to the language as defined in K&R. Many changes have become common across a wide variety of C implementations.

Many of the current implementations of C translators have established slightly stricter rules for combining objects of different types in C expressions. This change is also reflected in the standard. For example, integers (with the exception of 0) can no longer be assigned to a pointer variable unless an explicit cast is supplied. Listing 1 shows an example of how this requirement affects the writing of code.

The standard also follows common, current practice by extending the use of the unsigned type modifier to *char*, *short*, and *long* variables (K&R only defined unsigned *int*).

The requirement that names of members of structures and unions be unique across the entire program has been lifted. Now each structure defines its own name space, which means that member names may be reused in different structures without conflict.

Structures assignment as well as passing structures as arguments and having

```
func1()
{
int addr=1234;
char * cp;

cp = addr;              /* incorrect */
cp = (char *) addr;  /* correct */
```

Listing 1.

```
struct  firstst
        {
        int   f1;
        long  s1;
        };

struct  simplst
        {
        int   s1;         /* no name conflict with firstst.s1 */
        short s2;
        };

struct simplst  str1, str2;

struct simplst stfunc();          /*declare a function returning a structure */
func()
{
str2 = str1;          /* structure assignment */
str1 = stfunc(str2);  /* function with a structure argument, */
                      /*   returning a structure value */
```

Listing 2.

functions which return structures are also included in the standard. Listing 2 demonstrates how code that uses these features would look. Note that the assignment of the structure variable *str1* to *str2* causes the contents of *str1* to be copied into *str2*. This is not just a copying of pointers to the structures. Future modifications to members of *str1* will have no effect on the contents of *str2*.

The type *void* has been included to allow the specification that a function does not return a value (i.e., is a procedure). This allows a programmer to be more precise in specifying the correct use of a function.

The listing below demonstrates several uses of *void*. The declaration of the function *vfunc* uses *void* to specify that this is a function that returns no value. This is contrasted with the declarations of *lfunc*, which returns a long *int* and *ifunc* which returns an *int* (by default).

```
void vfunc();
long int lfunc();
ifunc ();
main()
{
vfunc();
(void) lfunc();
```

The main program in this example invokes the *void* typed function *vfunc* as a procedure that returns no value. The use of *void* in a cast is shown in the last statement in *main*. Here the *void* cast is used to show that the long value returned by *lfunc* is being explicitly ignored after this call. While the use of the cast in this situation is not required by the standard, it does allow for the writing of programs that are clearer and more maintainable.

Simple enumerated types have also been included. It should be noted that the enumerated types of C are significantly different than those supplied in some other languages, notably Pascal. Enumerated type variables are treated by C as simple integer variables, and no predecessor or successor functions are supplied. It is also possible to control the values of each of elements in the enumeration list. For example, in Listing 3 the enumeration constants *apple*, *grape*, *pear*, and *orange* have the values 0, 1, 4, and 5, respectively. The assignment causes *myjuice* to have a value of 4.

The last line of Listing 3 emphasizes the integer nature of enumerated type variables. The variable *myjuice* is decremented by one (using the -- operator). *Myjuice* now contains the value 3, not the value 1. No special processing is done to limit the values held by an enumerated type variable to those in its enumeration list.

The rules for identifiers have been changed slightly. Identifiers continue to be of an arbitrary length; however, internal identifiers (e.g., *auto* and *static* variables) may now be significant anywhere in the first 31 characters (as opposed to eight in K&R). External identifiers (e.g., global variables and non-static functions) are more restricted due to the large number of implementations that have assemblers, linkers, and other language-related utilities that can not support long names.

Conforming implementations are only required to support external identifiers (global variables and non-static functions), which are single case and significant in the first six characters. Implementations can, of course, allow a greater number of significant characters in an external identifier.

The committee has also clarified the rules in several areas where various implementations had taken different approaches. One area that falls into this category is the "writability" of strings. Some current implementations view strings as constants that cannot be modified (and put into ROM, if useful). Others view them as an array of characters that can be modified just as any array can be.

The current draft of the standard specifies that strings are constants and are not writable. It is allowable for a translator to supply writable strings as an extension to the language, however, portable programs should not use this extension.

Considerable variety in implementations has occurred in the area of macros. A fairly common practice in this area is "pasting" together tokens by eliding a comment between them. This practice, shown in Listing 4, is based on the assumption that a comment can be replaced by nothing rather than white space as specified in K&R. This practice will not have the desired merging effect in standards conforming translators since the standard explicitly states that comments are replaced by white space (e.g., a blank).

Since the merging facility was viewed as useful, the committee defined a new, consistent syntax for token pasting on *#define* lines. This syntax uses a new *##* operator (Listing 5).

New rules and extensions

In addition to the noted changes as a result of common existing practice and clarifications, the committee has established some new rules and added several extensions to the C language to meet specific well-defined needs.

The most ambitious extension has been the addition of a facility, called function prototypes, which allows the checking and conversion of function call arguments. This feature has been added to address the C language weakness that there was no way to ensure that actual function arguments were correct other

```
func2 ()
{
enum fruit {apple, grape, pear=4, orange};
enum fruit myjuice;

myjuice = pear;
--myjuice;
```

Listing 3.

```
Old style of merging

/* merge arguments a and b into a single token */
#define concat(a,b)   a/**/b
```

Listing 4.

```
New style of merging

/* merge arguments a and b into a single token */
#define concat(a,b)    a # b
```

Listing 5.

```
int proto1 (long, char *, short);
double proto2 (void);
unsigned int oldfunc();

func()
{
int     abc;
char *  cp;
short   xyz;
double  dbl;

abc = proto1(abc, cp, xyz); /* abc is widened to a long */
dbl = proto2(abc);          /* invalid function call */
```

Listing 6.

```
main()
{
unsigned char uc;
signed char   sc;
char          c;
int           abc;

uc = '\377';
sc = '\377';
c = '\377';
              /* assuming 8-bit char and two's complement/*
abc = uc;     /* abc becomes 255 */
abc = sc;     /* abc becomes -1 */
abc = c;      /* depends on the implementation, */
              /*  either 255 or -1 */
```

Listing 7.

than by tedious hand-checking. Some of the most difficult-to-find bugs are related to mismatched types between a function's parameters and the actual arguments on the function call.

The optional prototype facility allows the specification of a function declaration that includes type information for each of the parameters to the function. When present, it causes the translator to check each of the actual arguments on a function call. If the arguments match the corresponding parameter type in the prototype then nothing is done.

If an actual argument has a different type, but is convertible to the correct type by the rules of assignment, then the code to do the appropriate conversion is added. For example, an *int* actual argument would be converted to a *long* if the corresponding argument in the prototype was a *long*. If the arguments are different and there is no valid assignment conversion (e.g., between *int* and structure type) then an error has occurred.

Listing 6 demonstrates the declaration for several functions. *Proto1* is declared to be a function that returns *int*. It has three parameters which have types *long*, pointer to *char*, and *short*, respectively. *Proto2* is declared to be a function, returning a double. It has no parameters (note the use of the *void* keyword). *Oldfunc* is defined as a function that returns unsigned *short*. This declaration does not contain any parameter type information, and therefore the traditional rules that did not include any checking or conversion apply on the calls to *oldfunc*.

The function calls made in *func* will be handled as follows. The call to *proto1* will be checked. It will be determined that *abc* is of type *int* and that the prototype calls for a *long*. The appropriate conversion code will be generated. The other arguments match, therefore no additional special work is required. The call to *proto2* will be checked, and since the prototype says there are no arguments and the call has an argument, the call is invalid.

There are several places in C where the sign of a value is allowed to be whatever is fastest for the compiler to generate. This is seen when *char* and bit field values are widened to integers during expression evaluations. In these cases it is implementation-defined whether a value

that has the high-order bit set results in a positive or negative integer value.

This freedom for implementors allows for efficient code generation but causes problems when the signedness of a *char* or bit field value is important when writing portable programs. This problem has been addressed with the addition of the type modifier, *signed*, to the language.

The *signed* type modifier guarantees that the translator will view the bit pattern in the *char* variable or bit field as representing a signed value. Listing 7 shows an example of the three types of signedness that *char* variables can take on. Assuming an 8-bit *char*, the assignment of "\377" (octal 377) will set all of the bits in each of the variables.

However, when these variables are used (e.g., widened during the assignment to an integer variable), they are interpreted in different ways. The *unsigned char* is guaranteed to take on only zero and positive values. The *signed char* includes both positive and negative values. The "plain" *char* has an implementation-defined range. However, since implementations are allowed to select the fastest widening available in the hardware, plain *char* is an appropriate choice when the signedness of the value is not important (e.g., when it will only hold printable characters).

Two new type modifiers have also been added to the C language: *const* and *volatile*. The *const* modifier has been added to allow the specification that a data item will not be changed during the course of the program. This information is particularly valuable in small environments where constant data could be placed in ROM, if desired.

The presence of the *const* modifier causes the compiler to check for the invalid direct assignment, incrementing, or decrementing of the *const* object. Indirect attempts to modify *const* objects via pointers cause undefined behavior that the compiler is not required to diagnose.

The code fragment in Listing 8 shows how definitions using the *const* type modifier would look. The first line declares a constant *xyz* with a value of 24. Next, the constant array *arr* is defined and initialized. *Const* objects should always be initialized since modifications are not allowed in the body of the code.

The *volatile* type modifier indicates to the compiler that this object may be modified in ways unknown to the translator. It prevents the translator from optimizing any expressions that use this object. This has been added to address the need, in many microprocessor environments, for the adequate handling of memory mapped I/O.

In a memory mapped I/O system, a read or write of a location may cause special side effects, even though the statement appears pointless or unnecessary from the C code point of view.

The code fragment that follows defines and uses the *volatile* object *iomem*. If *iomem* had not been volatile, the translator would have been free to eliminate the first assignment to *iomem* since there is no use of *iomem* before it is assigned a new value in the second statement. Since *iomem* is volatile, this possible optimization is not allowed. This preserves the special semantics that may be associated with repeated writes to memory mapped I/O location.

```
main ()
{
volatile char iomem;
iomem = 0;
iomem = 1;
```

An important benefit of the *volatile* modifier is that it frees the translator to do sophisticated optimizations when it is not present.

This previous discussion has been a brief overview of the language-related work which is currently going on in the standards effort for the C programming language. I have not presented all of the changes and features that people will find new and different in the standard. However, I have tried to cover some of the major areas of change and to explain the underlying goals and principles that have guided all of the changes.

People interested in becoming involved in the standards activity or in obtaining a copy of the working document which, when complete, will become the draft proposed standard should contact either:

Jim Brodie
Motorola Microsystems
2900 South Diablo Way
Tempe, Ariz. 85282
(602) 438-3456

or the committee vice chair:

Tom Plum
Plum Hall
1 Spruce Ave.
Cardiff, N.J. 08232
(609) 927-3770

Jim Brodie has an M.S. in computer science from Arizona State Univ., Tempe, Ariz. He currently works at Motorola in C compiler development. He convened the C Programming Language Standards Committee and is its chairman.

```
const short xyz = 24;
const int   arr[4] = {188, 996, 44, -17};

func ()
{
int abc;

abc = xyz + arr[3]; /* appropriate use of constants */
xyz = 10;           /* invalid use of a constant */
arr[2]++;           /* invalid use of a constant */
```

Listing 8.

C + UTILITY LIBRARY = C PRODUCT

C

- We have over 200 *complete, tested, and, documented functions.* All source code and demo programs are included.
- The library was specifically designed for software development on the IBM PC, XT, AT and compatibles. There are *no royalties.*
- Over 95% of the source code is written in C. *Experienced programmers* can easily "customize" functions. *Novices* can learn from the thorough comments.

We already have the functions You are about to write

Concentrate on software development—not writing functions.

THE C UTILITY LIBRARY includes:

● Best Screen Handling Available ● Windows ● Full Set of Color Graphics Functions ● Better String Handling Than Basic ● DOS Directory and File Management ● Execute Programs, DOS Commands and Batch Files ● Complete Keyboard Control ● Extensive Time/Date Processing ● Polled ASYNC Communications ● General DOS/BIOS gate ● And More ●

- The Library is compatible with: Lattice, Microsoft, Computer Innovations, Mark Williams and DeSmet.

C Compilers: Lattice C—$349, Computer Innovations C86—$329; Mark Williams C—$449.

C UTILITY LIBRARY $149

Order direct or through your dealer. *Specify compiler* when ordering. Add $4.00 shipping for UPS ground, $7.00 for UPS 2-day service. NJ residents add 6% sales tax. Master Card, Visa, check or P.O.

ESI

ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, New Jersey 07040 914/762-6605

CIRCLE 27 ON READER SERVICE CARD

C Instead of FORTRAN?

By Anthony Skjellum

One of the most prevalent topics for disagreement between programmers is the choice of programming language. Many articles discuss the relative merits and demerits of various languages, and some authors make invidious comparisons to show that their own favorite language is better than the rest.

While many of these articles are quite informative, they usually fail to bring out how the application affects the choice of language. For example, one would normally choose a data base language for data base management applications as opposed to a general-purpose language like BASIC or Pascal.

Through examples this article attempts to illustrate why the C programming language works well for scientific and engineering applications. Before beginning, some background comments are in order.

Traditionally, scientific and engineering applications have depended very heavily on FORTRAN as the source language. FORTRAN was introduced early in the computer age. It subsequently became widely available, and the language afforded some portability between different hardware. FORTRAN became popular with scientists and engineers.

For many of these programmers, the concept of programming begins and ends with FORTRAN. Unfortunately, FORTRAN did not provide a programming environment that was conducive to structured, modular programming.

Furthermore, the intense connection of intrinsic functions to the compiler often forced programmers to resort to complex programming tricks. Many programmers became experts on writing tricky programs instead of programs that could be easily maintained, enhanced, or understood.

In all fairness, FORTRAN has undergone some face-lifts to include structured programming additions. But many programmers avoid these new features because they are not supported uniformly on different machines. Thus, traditional FORTRAN 66 has remained the standard form for programs.

New scientists/engineers automatically learn FORTRAN because that is the language they'll encounter in industry. A sort of vicious circle exists. Part of the problem is that students taught FORTRAN in an engineering/scientific setting often don't learn to carefully distinguish their FORTRAN code from an underlying algorithm.

No computer language can force the user into a perfect mold. The importance of a structured computer language is that it should make modular programming a comfortable, natural effort. At the same time, it shouldn't be so rigid that the programmer must work around language features and restrictions.

C satisfies these requirements as a structured language. It has the additional advantage that it is widely available with essentially no dialects. The standard library defined for C varies somewhat from implementation to implementation, but a good level of software portability exists.

We now turn to illustrations of the convenience and versatility of C in our application area.

Learning to program in C

An important aspect of programming is the language learning phase. C turns out to be easy to learn, although advanced features involving pointer arithmetic often are not learned at first. Fortunately, quite a bit can be done with just a few keywords and knowledge of a minimum number of library functions. Table 1 contains a list of keywords and functions useful when first learning C.

The relative ease of learning C is attractive for engineering and science students. If necessary, rudiments of C programming can be included as part of a secondary or college semester course, and students can learn the language incrementally as they develop programs. (It is also worthwhile to require modular program design and good documentation as part of the programming task.)

If C programming is used with several courses in a curriculum, students can gradually build a considerable knowledge of application programming in their field without taking specific programming courses.

In certain instances, students should learn to deal with software/hardware interfacing as part of course work. In many circumstances, however, this is avoided because students don't have time to learn assembly language in an engineering course.

However, since C handles address calculations and binary logic with ease, a small assembly-level nucleus could be prepared by the instructor, with students developing the bulk of the code in C instead of in assembly language. This would allow students to get hands-on experience they would otherwise miss. Development would be much less painful and time-consuming. This approach is feasible in a majority of cases, and it can often be used in practical applications also.

With all these comments behind us, let's turn to some example programs. They are designed to convey some of the flavor of C.

Example no. 1

Imagine the following problem. We need to develop a bisection method algorithm that can be applied to several distinct real functions within a program. The bisection method is a robust means for finding zeros of functions. For example, if our function were

$$f(x) = x - 1.0$$

then there would be exactly one zero, namely when x is equal to one. In more interesting cases, we often can't deduce the answer by inspection. (For more specifics on this method, consult Figure 1.)

In such cases, we need a function to find the desired zero or zeros. Such a program is depicted in Listing 1. (All listings referred to in this article are available in electronic form. You can download them either from the *COMPUTER LANGUAGE* Bulletin Board Service—(415) 957-9370—or take them from *COMPUTER LANGUAGE*'s account on CompuServe by typing "GO CLM".)

Our function *_bisect()* will find zeros of the function called *f()*, which we would define elsewhere in our program. Some valid examples for *f()* are presented in Listing 2. An example call to *_bisect()* might be the routine in Listing 3.

As we defined our task, we'll want to use a bisection method on more than one function. For example, we might want to perform bisection on three functions: *f()*, *g()*, and *h()*. To do so, we could write three versions of the *_bisect()* function.

This would work but is unnecessary and also a repetition of effort. To avoid this, we make the function under evaluation an argument to a more general function called *bisect()*, which we present in Listing 4. A valid calling sequence to this new routine might be as presented in Listing 5.

The importance of this improvement should not be missed. We have generalized the function to the point where it can be applied like a mathematical operator. That is, the function to be worked on is part of the argument list. Thus, *bisect()* operates on the function specified over the interval selected.

C provides the ability to use functions as arguments through the concept of pointers to functions. The name of a function without parentheses is an address. So

Basic keywords and functions in C

Keywords	Use
for	Used for looping structures
break	Break from looping structures
continue	Continue at start of loop
if . . . else	Conditionals
return	Return a value from a function
int	Define integer type
float, double	Define real single, double precision types
Functions	**Use**
exit()	Unconditional program termination
printf()	Formatted print function
scanf()	Formatted input function

Table 1.

Description of the bisection method

Object

Find a solution of $f(x) = 0$ (to within a tolerance) given a continuous function $f(x)$ on the closed interval $[a,b]$. The values $f(a)$, $f(b)$ must have opposite signs so that a zero exists in the interval $[a,b]$.

Algorithm

```
Step 0:    See if f(a), f(b) have opposite signs if not, print error message and
           abort program
Step 1:    Compute c = (a+b)/2
Step 2:    Compute t = f(c)
Step 3:    If |t| < tolerance, we are done. Return c as the location of zero.
Step 4:    If t > 0 and f(b) > 0
                   b = c
           If t > 0 and f(b) < 0
                   a = c
           If t < 0 and f(a) < 0
                   a = c
           If t < 0 and f(a) > 0
                   b = c
           This is the bisection of the interval
Step 5:    Go back to step 2
```

Figure 1.

```
double _bisect(a,b,prec)        /* find a zero of f(x) */
double a,b;                     /* limits */
double prec;                    /* convergence precision */
{
        double fabs();   /* absolute value (standard function) */
        double f();      /* this function returns double */
        double curr;     /* current point */
        double temp_val; /* current value */
        int cond;        /* indicates condition of signs of problem */

        if((f(a) <= 0.0)&&(f(b) >= 0.0))
                cond = 0;       /* no reversal */
        else
        if((f(a) > 0.0)&&(f(b) < 0.0))
                cond = 1;       /* reversal*/
        else
        {
                printf("Illegal bisection request [%e,%e]\n",a,b);
                exit(1);
        }

        while(1)                        /* loop 'forever' */
        {
                curr     = (a+b)/2.0;   /* bisect interval */
                temp_val = f(curr);     /* get value there */

                if(fabs(temp_val) < prec) /* we are done */
                        break;

                if(temp_val < 0.0)
                {
                        if(!cond) a = curr;
                        else      b = curr;
                }
                else
                {
                        if(!cond) b = curr;
                        else      a = curr;
                }

        }/* end while(1) */
        return(curr);

} /* end _bisect() */
```

Listing 1.

the word "cos" specifies the address of the function *cos()*. The corresponding argument variable *fn* in the function *bisect()* is defined as follows:

```
double (*fn)();
```

This means that *fn* is a pointer to a function which returns double precision real numbers. Whenever we write *(*fn)(x)*, we actually end up calling the originally specified function, i.e., *cos(x)*.

This provision is useful throughout program development. It is one of the most useful concepts of C. We can define very general functions and pass other functions to them to define their actual behavior in a particular instance. Programmers not only avoid repetitious coding but the

```
double f(x)
double x;
{
        return(x - 1.0);                  /* f(x) = x - 1.0 */
}

double f(x)
double x;
{
        double sin();                     /* sin() function */
        return(sin(2.0*x));               /* f(x) = sin(2x) */
}

double f(x)
double x;
{
        double temp = 0.0;
        double power = 1.0;
        int i;

        for(i = 1;i < 6;i++)    /* i loops from 1 to 5 inclusive */
        {
                temp  = temp + power/((double)i));
                power = power * x;
        }

        return(temp);   /* f(x) = polynomial */
}
```

Listing 2.

```
double y;
...
y = _bisect(1.0,2.0,.0001); /*
                                range to search [1.0,2.0],
                                zero tolerance is .0001
                             */
```

Listing 3.

```
double bisect(fn,a,b,prec)      /* find zero of specified function */
double (*fn)();                 /* function to find zero of */
double a,b;                     /* limits */
double prec;                    /* convergence precision */
{
        double fabs();   /* absolute value (standard function) */
        double curr;     /* current point */
        double temp_val; /* current value */
        int cond;        /* indicates condition of signs of problem */

        if(((*fn)(a) <= 0.0)&&((*fn)(b) >= 0.0))
                cond = 0;       /* no reversal */
        else
        if(((*fn)(a) > 0.0)&&((*fn)(b) < 0.0))
                cond = 1;       /* reversal*/
        else
        {
                printf("Illegal bisection request [%e,%e]\n",a,b);
                exit(1);
        }

        while(1)                        /* loop 'forever' */
        {
                curr     = (a+b)/2.0;   /* bisect interval */
                temp_val = (*fn)(curr); /* get value there */

                if(fabs(temp_val) < prec) /* we are done */
                        break;

                if(temp_val < 0.0)
                {
                        if(!cond) a = curr;
                        else      b = curr;
                }
                else
                {
                        if(!cond) b = curr;
                        else      a = curr;
                }


        }/* end while(1) */
        return(curr);


} /* end bisect() */
```

Listing 4.

DeSmet C

8086/8088 Development Package $109

FULL DEVELOPMENT PACKAGE
- Full **K&R** C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete **STDIO** Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 AND SOFTWARE FLOATING POINT

OUTSTANDING PERFORMANCE
- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER $50
- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT $35
- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt. (35)

SHIP TO: ____________________

____

____________________ ZIP________

C WARE CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add $5, elsewhere add $15. Checks must be on US Bank and in US Dollars. Call 9 a.m. – 1 p.m. to CHARGE by VISA/MC/AMEX.

CIRCLE 18 ON READER SERVICE CARD

greater generality often makes the functions applicable for future purposes. Any greater effort in carefully coding such a function is rewarded by reduced effort on a future project.

In introducing C, pointers to functions should be discussed early. Students should be taught how to use the device; explanation of addresses and pointers can wait until arrays have been introduced and understood. In this context, the concept of an address can be more easily motivated.

In this first example, we have illustrated an algorithm in its C incarnation, and we also introduced the idea of passing functions as arguments. When building software for scientific/engineering applications, we are often faced with the need to perform operator-like procedures on functions we define. Often we have no analytical representation for the functions, so we must consider some sort of interpolation. The next example illustrates how we can turn numerical data into a "black box" and thus simplify other computations.

Example no. 2

As a result of experimentation or numerical computation we often obtain a set of N data points:

$$\{x[i], y[i]\} \qquad i = 1, 2, \ldots, N$$

From this data, we seek to obtain a continuous functional relation of the form $y = f(x)$. Since we only know the relation between x, y at N points, we have to interpolate (or perhaps extrapolate) to get a value we require. If we are successful, we will be able to bury the discrete nature of our data within a function that appears continuous. Once we do this, we can potentially use functions such as *bisect()*, which accept a continuous function as an argument.

The quality of answers that this technique produces depends heavily on the method of interpolation we use. For certain applications, linear interpolation suffices. Listing 6 is a general linear interpolation routine. This is the first step to placing our data in a black box.

Given the general linear interpolation routine, we can create a function that appears to other routines as if it were continuous. If an x is requested outside the known interval, the interpolator will detect this and return 1.0e300, which is nearly "infinity" in double precision.

Error handling must be performed at some level, but this is up to the programmer. Listing 7 provides the outer level function, which uses *linterp()* on data arrays called *xarray[]* and *yarray[]*. These arrays are defined to contain *npoint* points. We have been careful to make *linterp()* general so that it can deal with many different data arrays instead of just a specific pair. This makes life easy for the programmer. For example, inverse interpolation of the *xarray[]*, *yarray[]*, can be done without difficulty. To perform this new calculation, we need only change the line

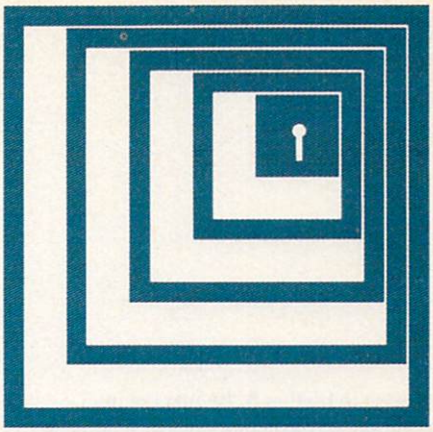
```
y = linterp(xarray,yarray,x,npoints);
```

of Listing 7 to the analogous form

```
x = linterp(yarray,xarray,y,npoints);
```

```
double cos(); /* use cos() function for calculation */
double y;     /* destination for answer */
...
y = bisect(cos,0.0,3.14159,.0001);
              /*
                 bisect cos() function
                 interval [0.0,3.14159]
                 zero tolerance .0001
              */
```

Listing 5.



```
/*

                           Linear Interpolation
*/

double linterp(xvalues,yvalues,x,points)
double xvalues[];               /* source of x values (sorted) */
double yvalues[];               /* source of y values (sorted) */
double x;                       /* find y given this x */
int points;                     /* number of points */
{
        int i;                  /* looping variable */
        double low_x,high_x;    /* used for tabular x values */
        double low_y,high_y;    /* used for tabular y values */
        double y;               /* temporary for result */

        if((x < xvalues[0]) || (x > xvalues[points-1]))
                return(1.0e300);        /* flag illegal result */

        for(i = 0;i < points;i++)
        {
                low_x  = xvalues[i];
                high_x = xvalues[i+1];

                if((low_x <= x) && (high_x >= x))
                {
                        low_y  = yvalues[i];
                        high_y = yvalues[i+1];
                        break;
                }
        }

        y = low_y + ((x-low_x)/(high_x-low_x))*(high_y-low_y);
                        /* formula for a straight line */

        return(y);
}
/*

*/
```

Listing 6.

as seen in Listing 8. By inverse interpolation we mean that given a *y* value, we can find the corresponding *x* (possibly not unique). Whether this is useful for specific data depends on the nature of the data and the task at hand. Normally we would have to re-sort the data before attempting this so that *y* values became ordered lowest to highest.

In this second example, we have shown how to build routines to most conveniently structure discrete data within a program. This approach avoids many complications when using general routines and will come in handy when performing many tasks in scientific/engineering applications. Usually more sophisticated interpolation will be employed. These examples were included to show the point, not to solve the general problem.

C can be used successfully for the type of applications considered in this article. Since it is readily available, powerful, and easy to learn, there is good reason to hope that many engineers and scientists will find C useful in future programming tasks.

Anthony Skjellum graduated from Caltech with a B.S. in physics in 1984. He is now pursuing graduate studies in chemical engineering, exploring the use of parallel computers and distributed processing for chemical engineering applications.

```
double f(x)             /* black box form for data */
double x;               /* point to evaluate at */
{
        double y;

        extern double xarray[],yarray[]; /* our data defined
                                            as external */
        extern int npoints;              /* points in arrays */

        y = linterp(xarray,yarray,x,npoints);
                        /* compute answer */

        return(y);      /* return answer (no error check) */
}
```

Listing 7.

```
double g(y)             /* inverse black box form for data */
double y;               /* point to evaluate at */
{
        double x;

        extern double xarray[],yarray[]; /* our data defined
                                            as external */
        extern int npoints;              /* points in arrays */

        x = linterp(yarray,xarray,y,npoints);
                        /* compute answer */

        return(x);      /* return answer (no error check) */
}
```

Listing 8.

The C Compiler Thousands Rely On

C-86™

NEW-IMPROVED Version 2.2 Compiles 25% Faster
IBM-PC AT Support

When the going gets tough, Optimizing C86 comes through time and time again. C86 is a highly dependable C compiler that has been optimized through the years to provide the best combination of reliability, speed, and performance.

FAST, IN-LINE 8087/80287 SUPPORT

Now you can take full advantage of 8087/80287 capabilities, allowing your programs to run many times faster than possible with other C compilers. Plus the source code to all routines is included, so you have complete control over all functions.

MORE OF THE FEATURES YOU WANT

- **SOURCE** is provided to all libraries for total programming control. The source includes a set of standard UNIX routines plus many DOS specific functions.
- **SPECIAL IBM-PC LIBRARY** including communication, screen, and keyboard handling functions.
- **COMPATIBLE WITH WIDELY AVAILABLE LIBRARIES** such as HALO screen graphics and many, many others (call for list).
- **TOPVIEW SUPPORT LIBRARY** provides windowing capabilities.
- **SPEED OPTIMIZATION** — there's always room to tighten your code, and Computer Innovations has the tools to help. For example, *PROFILER-86* helps identify key areas for optimization.

TECHNICAL SUPPORT, NOBODY DOES IT BETTER

Computer Innovations has earned a reputation for providing customer support that is **unequalled** in the industry. This includes a user's group, an on-line bulletin board, and a user's newsletter.

JOIN THE THOUSANDS OF PROGRAMMERS WHO TRUST AND RELY ON C86

For Further Information Call 800-922-0169.
Technical Assistance Call (201) 542-5920.

Computer Innovations features a full line of C products including **C-to-dBase** (dBase development tool) and **Introducing C** (C Interpreter Language Learning System). Call or write for a product profile.

COMPUTER
INNOVATIONS, INC.
980 Shrewsbury Avenue, Tinton Falls, NJ 07724

For Further Information Call
800-922-0169
Technical Assistance Call (201) 542-5920

© 1984 Computer Innovations, Inc.

CIRCLE 8 ON READER SERVICE CARD

An Open Letter
to
BDS C Users

Hi, I'm Leor Zolman, author of the **BDS C Compiler** and president of BD Software, Inc. I would like to take this opportunity to personally thank all of you out there who have purchased and used BDS C over the past five years for your overwhelming support, feedback, and ***especially*** your kind words about the package to potential new users. As you may have noticed, I've been doing a lot of advertising lately; while the advertising tells the world that BD Software, Inc. is still alive and selling C compilers, nothing has served to popularize the package as much as word of mouth, and for that I am grateful.

While I'm on the subject of expressing appreciation, someone who really deserves a bunch of it is Robert Ward, coordinator of the C User's Group. In the three-plus years that Robert has been in charge of the Group, he has done a remarkable job of newsletter production and public domain software acquisition. If any of you who use BDS C are not already enrolled as members of the C User's Group, then you are missing out on some really ***spectacular*** sofware available for peanuts. Here are some of the offerings: a package that effectively makes BDS C put out assembly language, screen editors, text formatters, a bulletin board system, cross-assemblers, Unix-style libraries (if you don't like the standard CP/M-specific BDS C library), and lots more. Robert, you done good.

BDS C first went on sale in August of 1979. For the first several years of its existence, all copies were vended through one particular exclusive distributor in New York. Unfortunately, most dealers and many users had trouble dealing with that distributor (to put it mildly), and many dealers probably wrote off BDS C due to the difficulty inherent in obtaining it. Late in 1982 I decided to market BDS C myself, and therefore began to interact directly with a number of software dealers and distributors. A number of those dealers and distributors have been a special pleasure to work with, and I would like to list them here so that any authors out there considering the choice of distributors might benefit from my experience: **New Generation Systesms, The Programmer's Shop, Westico, Workman and Associates.** In Japan: **Lifeboat Inc.** (not to be confused with Lifeboat New York!) In Australia: **ARCOM Pacific.**

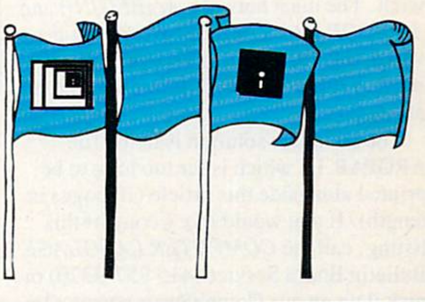
Thanks again to BDS C users, distributors and Robert Ward for a fun-filled, productive five year adventure. If the next five years prove to be even half as gratifying as the last, I'm going to be one lucky programmer...

BD Software, Inc.

P.O. Box 2368
Cambridge, Ma. 02238
(617) 576-3828

Part I—Parsing command line arguments

Programming Macros in C



By Alexander B. Abacus

Part I of this article will cover macros as building blocks for parsing command line arguments in C programs. Next month we'll explore how macros can also be used as effective tools when debugging large programs. Finally, in Part III of this special series, we'll look at how to use macros in C to translate a common representation of data into different but related tables required by different programs.

Most programs take some arguments from the command line. It is desirable to have a standard way of parsing those arguments to insure uniformity of the user interface and reduce programming effort spent on this common task.

By careful selection of macros, argument parsing can be specified in a very compact manner that resembles data declarations rather than executable code. Most error conditions are handled in the code generated by macro calls. The same degree of compactness cannot be achieved by using only functions without macros.

The parsing algorithm can be customized by setting values of several key variables. To reduce the volume of information the programmer has to specify, default values are provided whenever possible. The programmer can override each default. A separate macro sets each key variable. Each macro has at most one argument and the name of the macro is mnemonically related to the effect it produces.

Command line arguments

Command line arguments are divided into positional arguments and key-letter arguments (also called options). Key-letter arguments start with a trigger character, which is usually a minus sign. The order of processing positional and key-letter arguments can be varied:

- Deferred processing of positional arguments. When all key-letter arguments are processed before any of the positional arguments, all key-letter arguments apply to all positional arguments regardless of the placement of key-letter arguments.
- Immediate processing of positional arguments. When all arguments are processed in order, a key-letter argument is effective only for the positional arguments that follow after it on the command line.
- Key-letter arguments must precede all positional arguments— required by many programs written in C. After detecting the first positional argument, these programs assume that all subsequent arguments are positional. This approach is popular because it is easy to implement. It is supported by the implementation described here but is not recommended.

Key-letter arguments are further divided into flag and text arguments. A flag argument consists of a key letter following a trigger character. Several flag key letters can be grouped together following one trigger character as in:

```
–ab
–a  –b
```

A text argument consists of a single key letter immediately following a trigger character and followed by a character string either immediately or after a white space, as:

```
–t Word
–t "Quoted word"
–tTail
```

Handling of text arguments in C programs is inconsistent. Some require white space after the flag letter, others do not allow it, while most programs accept both variations.

The input to the C program is the command line already parsed into words. A word is a sequence of characters between white spaces. White spaces can be part of a word only if the word is enclosed in quotes.

It is obvious that words passed to the C program do not correspond to our key-letter arguments. In the first example under flag arguments, there are two flag arguments but only one word is passed to the C program. In the first two examples under text arguments, there is one text argument but two words are passed to the C program.

Solutions

The following macro definitions provide tools for parsing positional and flag arguments. Text arguments, additional error checking, and some bells and whistles will be added in the complete solution.

The macros defined in Listing 1 are used as a "sandwich." Macros *ArgBEGIN()*, *ArgLOOP*, and *ArgEND* should be thought of as bread in the sand-

wich. The meat between *ArgBEGIN()* and *ArgLOOP* are the macros overriding provided defaults. Macros between *ArgLOOP* and *ArgEND* define key-letter arguments.

The complete solution is in the file ARGPAR.H, which is far too long to be printed alongside this article (10 pages in length). If you would like a copy of this listing, call the *COMPUTER LANGUAGE* Bulletin Board Service (415 957-9370) or pick it up on our CompuServe account by typing "GO CLM". If you have no access to a modem, send a stamped, self-addressed envelope to the editor and he'll mail you a copy. I strongly encourage you to take this extra step if you're truly interested in further pursuing this subject.

The following macros were selected from the file ARGPAR.H to illustrate the more important and often fundamental ideas on how to effectively use macros in parsing command arguments in C.

ArgBEGIN(ArgumentCounter, ArgumentVector)

Macro *ArgBEGIN()* starts the sandwich that defines argument parsing. Arguments of this macro are typically the names of the arguments of the function *main()*.

ArgTrigger(TriggerCharacter)

Any character can be specified as trigger character. Default is '−'.

ArgMin(LowLimit)

specifies the minimum number of positional arguments that must be coded on the command line. If the macro is not coded, the check is not performed.

ArgMax(HighLimit)

specifies the maximum number of positional arguments that may be coded on the command line. If this macro is not coded, the check is not performed.

ArgDescription(StringArray)

The argument *StringArray* supplies the description of command arguments. It is printed when requested by the *−?* option or after any error message that terminates the parsing of arguments.

ArgPosCall(Function)

This macro turns on the immediate processing of positional arguments. The supplied function is called when a positional argument is recognized. At that moment only those key-letter arguments that precede that positional argument have been processed. (See section on function interface.)

ArgKeyLeading

This macro forces the requirement that all key-letter arguments must preceed all positional arguments.

ArgLOOP

Macro *ArgLOOP* divides the *ArgBEGIN()—ArgEND* sandwich in two parts. The first part is optional and may be empty. If present it sets the modes that apply to argument parsing. All modes have default values provided by *ArgBEGIN()*. Each default can be overridden separately by coding a macro between *ArgBEGIN()* and *ArgLOOP*. Those macros have already been described.

The second part defines key-letter arguments. Macros defining key-letter arguments must be coded between macros *ArgLOOP* and *ArgEND*. One (and only one) such macro must be coded for each key-letter argument. Macros defining key-letter arguments are described next.

ArgFlagSet(KeyLetter, CounterVariable)

Character *KeyLetter* specified in this macro can be used as a flag. This macro increments the *CounterVariable* each time that flag is found in command line arguments. *CounterVariable* should be initialized to zero. Note that the flag is allowed to occur more than once on the command line.

ArgTextSet(KeyLetter, PointerVariable)

Character *KeyLetter* specified in this macro can be used to introduce text argument. *PointerVariable* is set to point to the first character of the text string. *PointerVariable* should be initialized to NULL pointer. Multiple occurrences of the text argument specified by this macro are not allowed.

ArgFlagCall(KeyLetter, FlagFunction)

Character *KeyLetter* specified in this macro can be used as a flag. This macro invokes *FlagFunction*. (See section on function interface.)

ArgTextCall(KeyLetter, TextFunction)

Character *KeyLetter* specified in this macro can be used to introduce text argument. This macro invokes *TextFunction*. (See section on function interface.)

ArgEND

Macro *ArgEND* ends the sandwich that defines argument parsing. After the execution of the *ArgBEGIN()— ArgEND* sandwich, *ArgumentCounter* and *ArgumentVector* will be updated to reflect only positional arguments.

Function interface

Functions required as the second argument of macros *ArgFlagCall()*, *ArgTextCall()*, and the only argument of the macro *ArgPosCall()* all have the same interface. They should be defined as:

```
char * name(keyLetter, textString)
char   keyLetter;
char * textString;
{...}
```

Character *keyLetter* contains the key letter for key-letter arguments and '\0' for positional arguments.

Pointer *textString* points to the text of the argument for positional arguments and key-letter text arguments and is a NULL pointer for key-letter flag arguments.

```
Simplified argument parsing, positional and flag

#define ArgBEGIN(A_argc, A_argv) \
   { /* Begin block with local variables. Ends with ArgEND. */ \
     struct \
     { int     argc, *ptrArgCount; \
       char ** argVector; \
       char *  (*ptrFunction) (); \
       char    keyTrigger, *keyPointer; \
       int     argIndex, charIndex, nextPositional, keysTerminated; \
     } a0; \
     a0.ptrArgCount = &(A_argc); a0.argVector = (A_argv); \
     a0.argc = *a0.ptrArgCount; a0.keyTrigger = '-'; \
     a0.ptrFunction = argPos;

#define ArgPosCall(A_Function) a0.ptrFunction = (A_Function);
#define ArgTrigger(A_Trigger)  a0.keyTrigger = (A_Trigger);

#define ArgLOOP \
     a0.keysTerminated = 0; a0.nextPositional = 1; \
     for ( (a0.argIndex = 1); (a0.argIndex < a0.argc); (++a0.argIndex) ) \
     { /* for all arguments */ \
       if ( ( a0.argVector[a0.argIndex][0] != a0.keyTrigger ) \
       ||    ( a0.keysTerminated != 0                         ) \
       ||    ( a0.argVector[a0.argIndex][1] == '\0'           ) ) \
       { /* not a key: bubble up positional arguments */ \
         a0.argVector[a0.nextPositional] = a0.argVector[a0.argIndex]; \
         (void) (*a0.ptrFunction) ( '\0', a0.argVector[a0.nextPositional] ); \
         ++a0.nextPositional; \
       } \
       else /* it is a key */ \
       { \
         for \
         ( (a0.charIndex = 1, a0.keyPointer = & a0.argVector[a0.argIndex][1]) \
         ; (   (  a0.keyPointer  != (char *) 0 ) \
           && ( (*a0.keyPointer) != '\0' ) \
           ) \
         ; (++a0.charIndex, ++a0.keyPointer) \
         ) \
         { /* for all characters in an argument */ \
           switch ( a0.keyPointer[0] ) \
           {

#define ArgFlagSet(A_KeyLetter, A_FlagCounter) \
           case A_KeyLetter: ++A_FlagCounter; break;
```

Listing 1. *(Continued on following page).*

```
#define ArgFlagCall(A_KeyLetter, A_Function) \
           case A_KeyLetter: (A_Function)( *a0.keyPointer, 0 ); break;

#define ArgEND \
           default: \
             (void) fprintf \
             ( stderr \
             , " Stop. Key letter \'%c\' rejected.\n" \
             , *a0.keyPointer \
             ); \
             exit (1); \
           } /* switch */ \
         } /* for all characters in argument */ \
       } /* if key or not a key */ \
     } /* for all arguments */ \
     (* a0.ptrArgCount) = a0.nextPositional - 1; \
   } /* End block begun with ArgBEGIN. */ \

static char * argPos( a_keyChar, a_argString )
char   a_keyChar, *a_argString;
{ return ( (char *) 0 ); }

/* ------------------------------------------------------------- */

#if defined DriverH

#include <stdio.h>

void main(argCount, argVector)
int    argCount;
char * argVector[];
{
  int    f_flag = 0;

  ArgBEGIN    (argCount, argVector)
  ArgPosCall  (argPos)
  ArgTrigger  ('-')
  ArgLOOP
  ArgFlagSet  ('f', f_flag)
  ArgFlagCall ('F', argPos)
  ArgEND

  exit (0);
} /* main */

#endif DriverH
```

Listing 1. *(Continued from preceding page).*

The returned pointer should be NULL unless an error has been detected.

Predefined flags

The following key-letters are predefined as flags:

−? causes output of argument description. (See macro *ArgDescription()*.)

−! enables all debugging code if the program is compiled with debugging code. This flag must be the first argument after the command name. Besides acting as a flag, it can also be immediately followed by text, without a separating white space. Text is used to enable selected parts of debugging code.

The flag initializes the global variables used by the debugging macros, to be described in Part II.

−− A trigger character followed by another trigger character signals the end of key-letter arguments. Remaining arguments are treated as positional even if they start with a trigger character.

− If the trigger character stands alone it is passed as a positional argument. This is frequently used as notation for standard input.

The test driver at the end of the include file shows how to code the sandwich of macros that have been described. Counter variables referenced as arguments of macro *ArgFlagSet()* should be initialized to zero. Pointer variables referenced as arguments of macro *ArgTextSet()* should be initialized to a NULL pointer.

This is just the tip of the iceberg on how macros can be used to improve C program development efficiency. Next month we'll focus on how to use macros in order to set up a systematic method for debugging your code *before* you're ready to compile.

Reference

Harbison, Samual P. and Steele, Guy L. *C: A Reference Manual.* Prentice-Hall, 1984, pp.26-84.

Alexander Abacus has a B.S. in electrical engineering and is a software consultant for CGA Computer Inc., Cranford, N.J. His software experience includes work for Sperry, IBM, and AT&T Bell Laboratories.

Debugging Bugging You?

Torpedo program crashes and debugging delays with debugging dynamite for the IBM PC ...

UP PERISCOPE!

First, you install the hardware.

The hardware's a special memory board that fits in a PC expansion slot. Its 16K of write-protected memory contains Periscope's resident symbolic debugger. No runaway program, however berserk it may be, can touch this memory!

Then you UP PERISCOPE.

Use Periscope's push-button break-out switch to interrupt a running program ... even when the system's hung! Periscope supports Assembly, BASIC, C and Pascal. In addition to the usual debugging capabilities, some of Periscope's features are:

Stop your system in its tracks at any time.

Use symbol names instead of addresses.

Run a program on one monitor and debug on another.

Monitor your program's execution with Periscope's comprehensive breakpoints.

Debug memory-resident programs.

Put your time to better use.

The Periscope system is $295. It carries a 30-day money-back guarantee and includes the memory board, remote break-out switch, debugger software, 100-page manual, and quick-reference card. The memory board is warranted for one year. A demonstration disk is $5.00.

System requirements for Periscope are an IBM PC, XT, AT or Compaq, PC-DOS, 64K RAM, 1 disk drive and an 80-column monitor. For MasterCard and Visa orders only, call 800/421-5300 (ext. R96) 24 hours a day. For additional information, call 404/256-3860 from 9 AM to 5 PM Eastern Time.

Get your programs up and running;

UP PERISCOPE!

Data Base Decisions / 14 Bonnie Lane /Atlanta, GA 30328

CIRCLE 30 ON READER SERVICE CARD

POWER TOOLS

from Phoenix

You are a serious programmer. Your income depends on developing software solutions for complex problems. Fast. Error free. You need powerful tools for your compiler. That's what we sell.

PLINK ™ 86

OVERLAY LINKAGE EDITOR

- overlays for modules are automatically loaded during execution
- generates symbol tables for PFIX 86 PLUS
- accepts unlimited number of files
- capacity for over 35,000 symbol, commonblock and variable names
- change overlay structure without recompiling
- sorted memory map reports
- user defined width/height for reports
- define symbols as absolute addresses or offsets from other symbols
- overlay is user configurable

List: $395

PFIX ™ 86 PLUS

DYNAMIC SYMBOLIC DEBUGGER

- debugs PLINK 86 overlays
- view source code while debugging
- set temporary or permanent breakpoints
- advanced tracing and trapping capability
- in-line assembler for temporary patches
- preserves program screen and supports dual CRT configurations
- multiple windows to view code, data, breakpoints, register/stack contents, simultaneously
- supports 8087 numeric coprocessor

List: $395

PMATE ™

TEXT EDITOR PROCESSOR

- automatic disk buffering
- 10 auxiliary buffers that can be edited
- garbage stack
- user customization
- 120 command macro language
- store macros permanently
- arithmetic and logical operations
- 100 numeric variables
- automatic formatting with indent and word wrap
- insert, overwrite and command modes
- horizontal/vertical scrolling

List: $225

[PC]² We Sell Power

Phoenix Computer Products Corporation
1416 Providence Hwy., Suite 220
Norwood, MA 02062
800-344-7200 (Mass.) 617-762-5030
Visa • Mastercard • American Express

CIRCLE 82 ON READER SERVICE CARD

C to Assembly Interface

By Brian H. Burger

The fundamental question that arises when confronted with a new general purpose programming language is "why use it?", which really means "what advantages does it have over other languages?"

Of course, the answer depends upon the type of programming tasks to be carried out and their complexity or magnitude. For complex programming tasks, any advantages will have considerable bearing on both the effort required to accomplish the task and the quality of the resultant program.

It is necessary to access architecture-dependent machine capabilities to efficiently accomplish most complex programming tasks. Even without considering efficiency this is readily apparent for systems software. Since it is the nature of any non-machine language to be independent of these architecturally dependent features, a mechanism is necessary to access these features. The relative ease and efficiency of reaching these features is a consideration in choosing a language.

In this article we will briefly look at the advantages of C over other compiled programming languages and then focus on how C can be easily interfaced to access architecture-dependent capabilities. In particular, we will show how Lattice C can be efficiently interfaced to IBM MACRO Assembler to access the IBM PC ROM interrupt system by providing source code to perform video I/O. The code provided can be extended to access other architecture-dependent features on the IBM PC or on any computer using an 8088/86 architecture.

High-level vs. assembler

But if one must access these architecture-dependent features in any case, why not write the entire program in the assembler language of the target computer? After all, assembler code should result in the most efficient object code.

The problem is that assembler code is the most time-consuming to write and understand after it is written and will result in a large amount of source code to monitor and maintain. In addition, the program will only work on the target computer (or processor) for which it is designed.

It is generally far superior to write programs in a high-level language for increased productivity and portability. Thus, the better high-level language is the one that is most productive and most portable since that is the reason for using it in the first place.

Yet the resultant program must still use computer resources efficiently. There is no advantage to using a language that produces a program too slow or too memory hungry, even if it can be generated quickly and will work on lots of machines—except perhaps when that program is used, not as the end result, but as part of the process of development.

A superior language generates a more resource-efficient program. In fact, during the early development of FORTRAN the efficiency of the program resulting from the language was considered to be of much greater importance than the design of the language itself. John Backus wrote in *The History of Programming Languages*:[1]

> *To this day I believe that our emphasis on object program efficiency rather than on language design was basically correct. I believe that had we failed to produce efficient programs, the widespread use of languages like FORTRAN would have been seriously delayed. In fact, I believe that we are in a similar, but unrecognized, situation today (1978): in spite of all the fuss that has been made over myriad language details, current conventional languages are still very weak programming aids, and far more powerful languages would be in use today if anyone had found a way to make them run with adequate efficiency.*

Separately compiled subprograms

A high-level language is used instead of assembler when the benefits of increased productivity and portablity outweigh the penalty of less efficient use of computer resources. So the best high-level language to use is the one which is most productive, most portable, and generates the most efficient object code.

Another feature of a superior language is its ease of use and, by implication, programmer productivity. A language is easier to use if one can separately compile any number of subprograms and link them together later to produce an executable program. A language that allows separately compiled subprograms is superior

```
T ENDS
;
DSPIO   PROC   NEAR
        POP    SI          ; SAVE RETURN ADDRESS
        POP    AX          ; GET PARAMETERS
        POP    BX
        POP    CX
        POP    DX
        POP    DI          ; OPTIONAL POINTER TO 4 WORD AREA
        INT    10H
        CMP    SP,BP       ; BP IS ALWAYS >= OFFSET OF LAST ARGUMENT
        JA     DONE        ; IF NO RETURN POINTER PASTITLE DSPIO -
                           ; SUBROUTINE TO PERFORM DIRECT SCREEN I/O
;
; int ax, bx, cx, dx ;
; char retval [8] ;
;
;   dspio( ax, bx, cx, dx, retval ) ;
;   dspio( ax ) ;
;   dspio( ax, bx ) ;
;   dspio( ax, bx, cx ) ;
;   dspio( ax, bx, cx, dx ) ;
;
;    ax - dx VALUES TO BE PLACED IN THE REGISTERS OF THE SAME NAME
;            AS NEEDED BY PARTICULAR VIDEO I/O FUNCTION
;
;    retval, IF SUPPLIED IS A POINTER TO A 4 WORD AREA INTO WHICH THE
;            4 REGISTERS AX - DX ARE COPIED AFTER RETURN FROM INT 10H.
;
DGROUP  GROUP  DATA
DATA    SEGMENT WORD PUBLIC 'DATA'
        ASSUME  DS:DGROUP
DATA    ENDS
;
PGROUP  GROUP  PROG
PROG    SEGMENT  BYTE  PUBLIC 'PROG'
        ASSUME   CS:PGROUP
        PUBLIC   DSPIO
;
RLAYOUT  STRUC          ; Return address space layout; pointed at by retval
RAX     DW    ?
RBX     DW    ?
RCX     DW    ?
RDX     DW    ?
RLAYOUT ENDS
;
DSPIO   PROC   NEAR
        POP    SI          ; SAVE RETURN ADDRESS
        POP    AX          ; GET PARAMETERS
        POP    BX
        POP    CX
        POP    DI          ; OPTIONAL POINTER TO 4 WORD AREA
        INT    10H
        CMP    SP,BP       ; BP IS ALWAYS >= OFFSET OF LAST ARGUMENT
        JA     DONE        ; IF NO RETURN POINTER PASSED, DON'T SAVE VALUES
        MOV    [DI].RAX,AX
        MOV    [DI].RBX,BX
```

Listing 1 *(Continued on a following page).*

to one that does not. It wastes computer time to recompile an entire program whenever a change is made, and it wastes programmer time to have to wait for a compilation before proceeding.

Without the capability for separate compilation of subprograms, it is extremely difficult to maintain a library of useful subprograms, next to impossible for more than one programmer to work on the program, and difficult—if not impossible—to access architecture-dependent capabilities. Certainly, useful programming techniques like information hiding, stubbing, modular design and development, and others are compatible with separate compilation of subprograms and are at least inconsistent with the idea of one monolithic program compiled as one unit.

C

The language that produces the most efficient object code, is most portable, allows for the separate compilation of subprograms, and is most productive to use is the best language. C is recognized as producing efficient object code because it deals with the same sort of objects that most computers do. Constructs like register-type variables, pointers, the increment/decrement operators (++, −−, +=, *=, etc.), bitwise logical operators (<<, >>, &, etc.), a macro capability, and others all lead to efficient use of machine resources.

C is considered to be a very portable language.[2] It allows and in fact encourages separately compiled subprograms. Part of its portability power and efficiency is a consequence of not providing language constructs to do architecture-dependent machine operations but instead to encourage the creation of standard libraries to accomplish these tasks.

The brevity and elegance of its expressive power leads to increased productivity. Note that I say "leads to." C requires an understanding of the elements of the computer and related concepts to a greater extent than many other high-level programming languages. The average COBOL programmer has little use for knowledge concerning the difficulties encountered when a data structure falls on an odd or even byte address boundary since COBOL assumes the programmer isn't concerned with these situations and consequentially accepts any resultant inefficiencies.

Since C doesn't provide any language constructs to accomplish architecture-dependent machine functions, it is particularly important that one can interface C efficiently to assembler to accomplish these tasks. Even though C compilers normally include a standard library of subprograms, some machine-dependent features are not always included. It is not practical to expect a language or compiler system to provide access to all or even most of these features since they are specific to a particular computer or architecture and frequently specific even to a particular configuration.

Architecture-dependent capability

But what is meant by accessing architecture-dependent machine capabilities? For example, machines frequently provide in ROM a set of routines for performing machine specific operations. On the IBM PC and other machines utilizing the Intel 8088/86 processors, these routines can be accessed in assembly language through the software initiated interrupt system.

An interface to an assembly language subprogram is therefore needed to access them from a high-level language. In another example, machines generally have multiple processors that work independently of the main processor from which they get instructions. In 8088/86 assembly, an instruction can be sent to one of these processors by using an assembly-level instruction named *OUT* designed for this purpose.

The basic problem then becomes interfacing from a high-level language to the assembly level to access these special features in an efficient and uncomplicated manner. The rest of this article is devoted to doing just that by illustrating the ease with which C can be interfaced to assembly to access the IBM PC video interrupt routines.

You may be wondering why we aren't interfacing to perform DOS calls instead, in particular the DOS version 2 extended screen and keyboard device driver. While doing this will likely make the program more portable for DOS machines, this DOS driver is slow and cumbersome. In any case, similar techniques can be used to interface to DOS interrupts.

Interfacing C

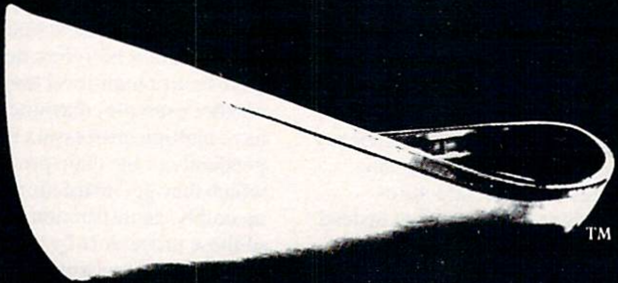
Specifically, Lattice/Microsoft C (referred to from this point forward as just C) is interfaced to IBM MACRO Assembler (MASM) to perform video I/O as shown in the MASM source code in Listing 1 and the C source code in Listing 2.[3,4] Figure 1 shows the C memory model that is being used. Its limitations are that the program can't be larger than 128K—the program segment is limited to 64K, and the combined data and stack segment is limited to 64K.

This discussion will not be concerned with C programs that use an extended memory model. If a program needs an instruction segment or data segment that exceeds 64K bytes, unavoidable inefficiencies creep into the implementation. There are many other ways to address this need rather than extending the memory model of the compiler. A discussion of this subject is beyond the scope of the present article. Furthermore, this model is insufficient in few cases.

Upon entry into a subprogram from C, the BP is the only general purpose register that needs to remain unmodified or be saved. C loads the BP with a pointer to the top of the stack before any arguments are pushed and uses that value to deallocate the stack upon return from the subprogram. This feature allows a variable number of arguments to be passed to the called subprogram, as the caller can deallocate the arguments from the stack by simply restoring the BP to the SP after the call. This mechanism requires a two-byte instruction (*MOV SP,BP*) to restore the stack after the call instead of the three- or four-byte instruction required to explicitly add a value to the stack pointer.

Since it is the caller's responsibility to deallocate arguments from the stack, there is no need to know at compile time the number of arguments to be passed. All other registers can be freely used except for the segment registers which must remain unchanged from their original

C Is The Language.
Lifeboat Is The Source.



*Lifeboat.*TM
The Leading Source And Authority For Serious Software.
1-800-847-7078.

In NY State: 212-860-0300

Serious Software For The C Programmer From Lifeboat.™

Lattice*® *C Compiler: *The serious software developer's first choice.*

Selected for use by IBM,® Texas Instruments, Wang,® MicroPro,® Ashton-Tate,™ IUS/Sorcim,® Microsoft® and Lotus™ to name a few of the many. Why?

Lattice C is clearly the finest 16 bit C compiler available today.

—Renowned for speed and code quality.
—Fully compatible with the C standards set forth by Kernighan and Ritchie.
—Four memory model options offer you unsurpassed control and versatility.
—Superior quality documentation.
—Now includes automatic sensing and use of the 8087 chip.
—Widest selection of supporting add-on packages.

Halo™: *A graphics development package rapidly emerging as the industry standard.*

—Over 150 graphics commands including line, arc, box, circle and ellipse primitives.
—New: multiple typefaces which may be scaled and rotated.
—Supports multiple viewports and is device independent.

C Food Smorgasbord™: *This beautifully written collection of C functions is a valuable time saver.*

—Library includes a binary coded decimal arithmetic package, level 0 I/O functions, a terminal independence package, IBM PC ROM BIOS access functions and much more.

Pmate™: *The premier editor for the programming professional.*

Pmate is a full screen editor with its own powerful macro command language:

—Perform on screen row and column arithmetic, alphabetize lists, translate code from one language to another, call up other macros.
—Customize Pmate almost any way you like.
—Contains 10 auxiliary buffers for storage of macros, text, subroutines.
—An "undo" feature allows the programmer to retrieve whole series of deleted items.

Additional C Tools Available From *Lifeboat:*

Panel™: Screen formatter and data entry aid.
Lattice Windows™: Windowing utility; create "Virtual Screens."
Plink-86™: The popular linker; includes extensive overlay capabilities.
Pfix86™: Dynamic debugging utility.
Pfix86 Plus™: Symbolic debugger with capacity to debug overlays.
Btrieve™: Database record access/retrieval library.
Phact: Multikeyed ISAM C-Function library.
Fabs: Fast access B-tree database function library.
Autosort: Fast sort/merge utility.
ES/P: 'C' program entry with automatic syntax checking and formatting.
Greenleaf Functions™: Library of over 200 popular C functions.
And much more.

YES! Please rush me the latest FREE Lifeboat™ catalog of C products.

*Company Name*____________________ *Business Phone*____________________
*Name*____________________ *Title*____________________
*Address*____
*City*________________ *State*__________ *Zip*________

Please check the category where Lifeboat can best help you:

☐ *Software development* ☐ *Corporate* ☐ *Education*
☐ *Dealer/distributor* ☐ *Government* ☐ *Other*__________

Call Direct: 1-800-847-7078 (In NY State: 212-860-0300)
Return coupon to: Lifeboat Associates™
1651 Third Avenue, New York, NY 10128.

Catalog ... 1984
C Programming Tools
Lifeboat
TM

C

© 1984 INTERSOFT CORPORATION

CIRCLE 64 ON READER SERVICE CARD

```
        MOV     [DI].RCX,CX
        MOV     [DI].RDX,DX
        JMP     SI
DONE:   PUSH    DI
        PUSH    DX          ; RESTORE STACK BECAUSE OF THE POSSIBILITY THAT THEY
        PUSH    CX          ; WERE NOT PASSED AS ARGUMENTS
        PUSH    BX
        JMP SI
DSPIO   ENDP
PROG    ENDS
        END
```

Listing 1 *(Continued from a preceding page).*

```
main()
{
 int retvals [4] ;
 int index ;
  SET_MODE ;
  hline(0,0,219,80) ;   /* draw box around screen */
  hline(24,0,219,80) ;  /* i.e., manual window using nonwindow characters */
  verline(0,0,219,24) ;
  verline(0,79,219,24) ;
  window(3,3,6,5) ;
  window(10,10,5,6) ;
  window(7,17,10,11) ;
  window(15,40,10,6) ;
  window(5,50,20,16) ;
      /* read c#define PAGE 0
 /* video modes */
#define s40x25_bw 0
#define s40x25_color 1
#define s80x25_bw 2
#define s80x25_color 3
#define med_color 4
#define med_bw 5
#define high_bw 6
 /* video functions */
#define set_type 256
#define set_cur 512
#define read_position 768
#define read_light_pen_position 1024
#define select_page 1280
#define scroll_up 1536
#define scroll_dn 1792
#define read_attribute_char 2048
#define write_attribute_char 2304
#define write_char 2560
#define set_palette 2816
#define write_dot 3072
#define read_dot 3328
#define write_teletype 3584
#define get_state 3840
 /* macros */
#define curpos(row,col) dspio(set_cur,PAGE,0,(((int)row)<<8)+col)
```

Listing 2 *(Continued on a following page).*

entry values. This mechanism is efficient and flexible—only one register needs to be saved upon call of a subprogram or remain unmodified, and it can be passed a varying number of arguments.

C passes arguments to its subprograms by value, not by reference. So any argument passed is a copy of the original value, not a reference or pointer to it. Arguments are passed by pushing them onto the stack in reverse order. That is, the first argument pushed is the rightmost argument appearing in the C function call. Upon entry to the subprogram, the first parameter appears on the top of the stack after the return instruction pointer pushed by the call.

The routine

The subprogram in Listing 1 is straightforward. To understand the particulars of the video I/O functions see the *IBM Personal Computer Technical Reference Manual.*[5] The macros in the source code in Listing 2 provide enough information to use most of the functions without the manual, and the source code preamble in Listing 3 provides most of the additional particulars needed.

The video I/O interrupt takes its arguments from the four general purpose registers AX through DX and returns values in those same registers. This subprogram takes five arguments corresponding to the four general purpose registers to be passed to the video I/O interrupt routine and a pointer argument used for storage of any return values. The return address is popped into SI, the arguments are popped off the stack directly into the four general purpose registers, and the fifth argument is popped into the DI register for later use. The second through fifth arguments are valid only if the subprogram was actually called with those arguments.

After the video I/O interrupt is executed, the subprogram determines if a pointer was actually passed as the fifth argument by recognizing that the BP is equal to the SP before any arguments were pushed. If it is a valid argument, the values of the four general purpose registers are copied to the locations pointed at by the fifth argument.

Similarly, all but the first argument are returned to the stack in case they were not actually passed. For those functions that return values in the general registers in addition to AX, all five arguments must be supplied. Otherwise incorrect values would get on the stack, and return values would be placed somewhere other than where they were supposed to go.

Only those arguments actually needed must be passed if no values are returned, plus any intervening arguments. When *dspio* is called to perform the set mode function in Listing 2, only one value is passed. Since the video I/O interrupt doesn't care what is in any of the registers it doesn't use when called, there is no need to pass trailing filler arguments.

However, if there are some intervening arguments, as when changing the cursor position in *curpos*, the intervening arguments must be included. To avoid the passing of filler arguments, the function code passed in the first argument can be used to branch to code that pops the arguments only into registers pertinent to the function. Thus, for function code 2 (i.e., *set mode*), it can be assumed that there will be no argument supplied for CX and the third argument can be popped into DX instead.

This is what *dispio* in Listing 3 is all about. *Dispio* avoids passing arguments unneccessarily, thereby increasing execution speed but for the penalty of increased instruction overhead.

An alternative

In Listing 3 a jump table is used to efficiently control the execution of code. It works by storing offsets in a table in such a way so that the function code stored in the first argument can be used as an index into the table to obtain the appropriate address to jump to. Thus, the order in which labels are stored into *CASE_TABLE* determines the code that will be executed upon receiving a function code in the high byte of the first argument. The first argument is copied into BX, the function code is extracted, multiplied by 2 (since addresses are words), and used as an index into *CASE_TABLE* to obtain the address to which the subprogram transfers control.

C treats subprograms as functions that return values. They return their function value to the caller in the general registers. The registers pertinent are determined by the type of the function. Thus, for *int*

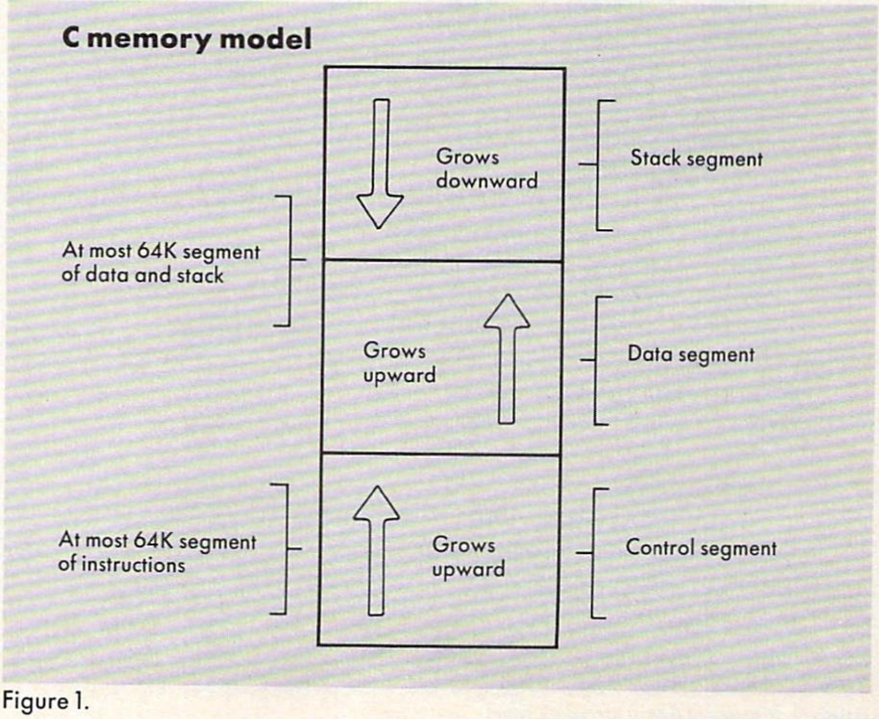


Figure 1.

```
#define SET_MODE dspio(s80x25_bw)
#define horline(ch,width) dspio(write_char+ch,PAGE,width)
#define hline(row,col,ch,width) curpos(row,col) ; horline(ch,width)
#define wrtchar(ch) dspio(write_char+ch,PAGE,1)

main()
{
 int retvals [4] ;
 int index ;
  SET_MODE ;
  hline(0,0,219,80) ;   /* draw box around screen */
  hline(24,0,219,80) ;  /* i.e., manual window using nonwindow characters */
  verline(0,0,219,24) ;
  verline(0,79,219,24) ;
  window(3,3,6,5) ;
  window(10,10,5,6) ;
  window(7,17,10,11) ;
  window(15,40,10,6) ;
  window(5,50,20,16) ;
      /* read cursor position */
  for (index=0; index <4; index++ ) {
    curpos(index,index+1) ;
    dspio(read_position,PAGE,0,0,retvals) ;
    printf("For page PAGE, Row:%u, Col:%u, cursor mode=%o,%o \n",
      retvals[3]>>8, (retvals[3]<<8)>>8, retvals[2]>>8,(retvals[2]<<8)>>8) ;
  }
     /* get current video state */
  dspio(get_state,PAGE,0,0,retvals) ;
  printf("Mode: %u, Columns: %u, Active Page: %u \n",
     (retvals[0]<<8)>>8, retvals[0]>>8, retvals[1]>>8 ) ;
}
window(row,col,width,depth) /* start at row,col, i.e. left upper corner */
char row,col ;
{
     /* first do 4 corners */
  curpos(row,col) ;
  wrtchar(214) ;
  curpos(row+depth,col) ;
  wrtchar(211) ;
  curpos(row,col+width-1) ;
  wrtchar(183) ;
  curpos(row+depth,col+width-1) ;
  wrtchar(189) ;
     /* then connect corners with horizontal and vertical lines */
  hline(row,col+1,196,width-2) ;
  hline(row+depth,col+1,196,width-2) ;
  verline(row+1,col,186,depth-1) ;
  verline(row+1,col+width-1,186,depth-1) ;
}
verline(row,col,ch,len)
char row, col, ch, len ;
{
  char cnt ;
     for (cnt=len; cnt > 0; cnt--) {
        curpos(row++,col) ;
        wrtchar(ch) ;
     }
}
```

Listing 2 *(Continued from a preceding page).*

functions the return value is stored in the AX register upon exit from the subprogram. The other registers are used for types requiring larger size return values.

For those video I/O functions that supply values upon return from interrupt in AX, no processing is required as they are passed back to the C caller as the called functions return value.

Thus, functions *read attribute and character* and *read dot* return information as the value of the *int* function *dispio* upon return. *Get state* returns all but the current active display page as *dispio*'s *int* function value in addition to storing it in the return area. *Get light pen position* returns whether or not the light pen switch is triggered as *dispio*'s *char* function value or as its high byte *int* function value, in addition to storing it.

In fact, all information can be returned as the function value of *dispio* upon return provided *dispio* is typed as double. However, C would have to generate code to deal with this long float type, which would result in unnecessary inefficiencies in most cases.

Efficiency

An instruction efficiency analysis of the minimum and maximum instruction sequences for both routines[6] shows that *dispio* is almost always faster. When no values are returned, it executes from 16% to 30% faster and when values are returned, from 24% faster to 5% slower.

Dispio is actually faster than the analysis indicates since it eliminates all excess pushing of arguments from C, while *dspio* requires pushing of filler arguments. For example, the worst case path for *dispio* is when it is doing a *read light pen* position and takes 198 cycles. *Dispio* requires two arguments from C and thus two pushes, while *dspio* would require five arguments from C or five pushes. Adding the extra three pushes for *dspio* yields 219 cycles compared with 198 cycles for *dispio*.

Clearly *dispio* will always be faster than *dspio*. If one is willing to put up with always supplying five arguments to *dspio*, the four pushes to restore the stack can be deleted, in which case it will be almost as fast as *dispio*. Of course, since C will be pushing arguments that aren't needed, that savings is partly an illusion.

Finally, one could return all values through C's function mechanism. The type of the function must then be chosen carefully, as the type declared will depend on what values are to be returned. To some, C opens a Pandora's box since C requires more knowledge of the machine and computer-related concepts than most other languages. To others, C allows increased productivity resulting from using a flexible high-level language with an acceptable efficiency penalty.

References

1. Backus, John. "The History of FORTRAN I, II, and III." *The History of Programming Languages*. Edited by Wexelblat, Richard L. Academic Press (1981): 25-45.
2. Harry, David. "An Implementation Demonstrating C Portability." *Computer Language* (Oct. 1984): 21-29.
3. *Microsoft/Lattice C Compiler*. Lattice/Lifeboat (1982), Microsoft (1983).
4. *IBM Personal Computer MACRO Assembler*. IBM (1981).
5. *IBM Personal Computer Technical Reference Manual*. (Aug. 1981): A-43 through A-45.
6. Rector, Russell, and Alexy, George. *The 8086 Book*. OSBORNE/McGraw-Hill (1980).

```
TITLE DISPIO - SUBROUTINE TO PERFORM DIRECT SCREEN I/O
;
;  #define MAX 8
;  char retvals [MAX] /* used by 3 functions */
;           /* (1) - read cursor position; retvals[0-1] = current cursor mode */
;           /*         retvals[3] = col, retvals[2] = row */
;           /* (2) - read light pen position  */
;           /*         retvals[0] = active(1)/inactive(0) */
;           /*         retvals[2-3] = pixel column, retvals[4]=raster line */
;           /*         retvals[6] = row, retvals[7] = col */
;           /* (3) - get state; retvals[0] = cols, retvals[1] = mode */
;           /*         retvals[2] = page */
;
;   setmode - dispio(mode) ;
;   set cursor type - dispio(set_curs_type,type) ;
;   set cursor position - dispio(set_cursor,page<<8,row<<8+col) ;
;   read cursor position - dispio(read_cursor_position,page<<8,retvals) ;
;   read light pen position - dispio(read_pen,retvals) ;
;   select active page - dispio(select_page+new_page_value) ;
;   scroll active page up - dispio(scroll_up+number_of_lines,page<<8,
;                                  lrow<<8+lcol, rrow<<8+rcol) ;
;   scroll active page down - dispio(scroll_down+number_of_lines,page<<8,
;                                    lrow<<8+lcol,rrow<<8+rcol) ;
;   get attribute character at current cursor pos - dispio(readac,page<<8) ;
;     returns attribute character as function value - high byte attribute,
;     low byte character
;   write attribute character at current cursor position - dispio(writeac+char,
;            page<<8+attribute, repeat count)
```

Listing 3 *(Continued on following page).*

```
;   write character - dispio(write_char+character,page<<8,repeat count) ;
;   set color palette - dispio(set_pallette,pallette<<8+color_value) ;
;   write dot - dispio(write_dot+color,col,row) ;
;   read dot - dispio(read_dot,col,row)
;              dot read returned as low byte of function value
;   write teletype - dispio(write_teletype+char,(page<<8)+color) ;
;   get state - dispio(get_state,retval)
;
DGROUP  GROUP  DATA
DATA    SEGMENT WORD PUBLIC 'DATA'
        ASSUME  DS:DGROUP
CASE_TABLE DW SETMODE,CURSTYPE,SETCURS,READCURS,READPEN,SETMODE
           DW SCROLL,SCROLL,READAC,WRITEAC,WRITEAC,READAC,WRITEDOT
           DW WRITEDOT,READAC,STATE
; case table uses 16 words
RLAYOUT  STRUC          ; Return address space layout
RAX     DW    ?
RBX     DW    ?
RCX     DW    ?
RDX     DW    ?
RLAYOUT ENDS
DATA    ENDS
;
PGROUP  GROUP  PROG
PROG    SEGMENT  BYTE  PUBLIC 'PROG'
        ASSUME   CS:PGROUP
        PUBLIC   DISPIO
```

Listing 3 *(Continued on following page).*

QUALITY SOFTWARE AT REASONABLE PRICES

CP/M Software by

Poor Person Software

Poor Person's Spooler **$49.95**
All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.

Poor Person's Spread Sheet **$29.95**
Flexible screen formats and BASIC-like language. Preprogrammed applications include Real Estate Evaluation.

Poor Person's Spelling Checker **$29.95**
Simple and fast! 33,000 word dictionary. Checks any CP/M text file.

aMAZEing Game **$29.95**
Arcade action for CP/M! Evade goblins and collect treasure.

Crossword Game **$39.95**
Teach spelling and build vocabulary. Fun and challenging.

Mailing Label Printer **$29.95**
Select and print labels in many formats.

Window System **$29.95**
Application control of independent virtual screens.

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add $5 handling charge. California residents include sales tax.

Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

CIRCLE 51 ON READER SERVICE CARD

The Tools You Need To C You Thru.

Now the *WizardWare*™ Applications Programmers Toolkit provides everything you need to increase your C programming productivity.

APT™ features:

- File Handlers
  - Direct Access
  - Keyed Access
- Generic Terminal Driver
- String Handling
- String Math
- Screen Generator
- Report Generator
- FIFO Que Routines
- Source Code
- Manual On Disk (45 pages)
- Tutorial On Disk (33 pages)
- Detailed Brochure on request

UNIX™ System V Version (available 1-1-85) . $995
MS-DOS™ Version $495
BDS C™ Version $395
Manual Only* $50

*Manual Cost will be applied to APT™ purchase price within 30 days ($10 re-stocking charge). U.S. funds only, please.

Trademark owners: UNIX™ (AT&T Bell Labs), MS-DOS™ (Microsoft, Inc.), BDS C™ (BD Software), Wizardware™ and APT™ (Shaw☆American Technologies)

Call: (800) 443-0100 Ext. 282
Ask for APT™ or Send Check To:
Shaw ☆ American Technologies
WizardWare™
830 South Second Street · Box 648
Louisville, KY 40201, U.S.A.
(C.O.D. and Foreign Orders - Add $5 Shipping/Handling)
References: Bank of Louisville, Citizens Fidelity Bank, Louisville Chamber of Commerce

VISA

CIRCLE 92 ON READER SERVICE CARD

```
;
DISPIO  PROC    NEAR
        POP     SI
        POP     AX
        MOV     BL,AH
        XOR     BH,BH
        CMP     BL,15           ; INVALID FUNCTION CODE ?
        JA      ERROR
        SHL     BX,1
        JMP     CASE_TABLE[BX]
;
ERROR:  MOV     AX,-1
        JMP     SI
;
SETCURS:   POP    BX
           POP    DX
SETMODE:   INT    10H
           JMP    SI
;
READCURS:  POP    BX
           INT    10H
           POP    DI
           MOV    [DI].RAX,CX
           MOV    [DI].RBX,DX
           JMP    SI
;
SCROLL:    POP    BX
WRITEDOT:  POP    CX
           POP    DX
           INT    10H
           JMP    SI
;
READAC:    POP    BX
           INT    10H
           JMP    SI
;
WRITEAC:   POP    BX
CURSTYPE:  POP    CX
           INT    10H
           JMP    SI
;
STATE:     INT    10H
           POP    DI
           MOV    [DI].RAX,AX
           MOV    [DI].RBX,BX
           JMP    SI
;
READPEN:   INT    10H
           POP    DI
           MOV    [DI].RAX,AX
           MOV    [DI].RBX,BX
           MOV    [DI].RCX,CX
           MOV    [DI].RDX,DX
           JMP    SI
DISPIO  ENDP
PROG    ENDS
        END
```

Listing 3 *(Continued from preceding page).*

C

Software Development

PCDOS/MSDOS

Complete C Compiler

- Full C per K&R
- Inline 8087 or Assembler Floating Point, Auto Select of 8087
- Full 1Mb Addressing for Code or Data
- Transcendental Functions
- ROMable Code
- Register Variables
- Supports Inline Assembler Code

MSDOS 1.1/2.0 Library Support

- All functions from K&R
- All DOS 2.0 Functions
- Auto Select of 1.1 or 2.0
- Program Chaining Using Exec
- Environment Available to Main

c-window™ Symbolic Debugger

- Source Code Display
- Variable Display & Alteration Using C Expressions
- Automatic Commands
- Multiple Breakpoints by Function & Line Number

8088/8086 Assembler

- FAST — Up to 4 times Faster than IBM Assembler
- Standard Intel Mnemonics
- Compatible with MSDOS Linker
- Supports Full Memory Model

8088 Software Development Package **$199.00**

Includes: C Compiler/Library, c-window, and Assembler, plus Source Code for c-systems Print Utility

c-systems

P.O. Box 3253
Fullerton, CA 92634
714-637-5362

CIRCLE 16 ON READER SERVICE CARD

Add EDITING to your Software with CSE Run-Time™

Your program can include all or a portion of the *C Screen Editor* (CSE).

CSE includes all of the basics of full screen editing plus source in C for only $75. For only $100 more get CSE Run-Time to cover the first 50 copies that you distribute.

Use capabilities like Full cursor control, block move, insert, search/replace or others. Portability is high for OSes, terminals, and source code.

Call for the "CSE Technical Description" and for licensing terms and restrictions. Versions for PCDOS, MSDOS, CPM, more.

Full Refund if not satisfied in first 30 days.

Call 800-821-2492

Solution Systems
335-L Washington Street
Norwell, MA 02061
617-659-1571

CIRCLE 93 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP™

helps compare evaluate and find products. Get answers.

SERVICE: FREE LITERATURE

One free call covers all programmer's software. Ask for a "Packet" on: "AI", BASIC, C, COBOL, Debuggers, Editors, FORTH, FORTRAN, Libraries, PASCAL, UNIX/PC or 30 "addons" for "C".

"C" Language

	OUR PRICE
MSDOS: C86 - 8087, reliable	call
Lattice 2.1 - improved - 30 addons	call
Microsoft C2.x	329
Williams - debugger	call
Instant C Interpreter, fast, full, debug	500
CPM80: Ecosoft C-now solid, full, faster	225
MAC: **Megamax** - fast, full, tight	295

EDITORS

	Runson	
BRIEF - Intuitive, flexible	PCDOS	195
PMATE - powerful	8086	195
VEDIT - full, liked	8086	119

ARTIFICIAL INTELLIGENCE

IQ LISP - full 1000K RAM	PCDOS	call
TLC LISP - with "classes", nice	MSDOS	250
MicroProlog - by Logic Prog. Assem.	MSDOS	285
PROLOG-86 - standard, Learn fast	MSDOS	125
EXSYS - Expert System	PCDOS	295

Feature

PERISCOPE DEBUGGER — "Reset Box", own RAM, Registers, symbols, line nums, 2 screen, PCDOS. $295.

Recent Discovery

Introducing-C : C Interpreter and training system. Nice. Thorough. PCDOS. Only $95.

FORTRAN

	Runson	OUR PRICE
MS Fortran - Improved	MSDOS	249
DR Fortran-86 - full '77'	8086	259
F77L - by Leahy - Nice.	MSDOS	449
RM Fortran	MSDOS	545

SUPPORT PRODUCTS

LIBRARIES: BTRIEVE ISAM	MSDOS	215
CIndex+ - ISAM, source, no royalt.	8086	375
CSHARP Realtime - source, full	MSDOS	600
CUtil by Essential	MSDOS	139
DATABURST - Screens-C, BAS	MSDOS	
GraphiC - 4200 x 3100, source	MSDOS	195
Greenleaf C - thorough	MSDOS	165
HALO Graphics - fast, full	PCDOS	175
TOOLS: MULTILINK - multitask	PCDOS	65
Polylibrarian-thorough	MSDOS	89
PolyMAKE-manage, compiles	PCDOS	89
Profiler-86-easy to setup, symbols	MSDOS	125
XENIX - "true S3", rich, + C-MSDOS	PC	1285

Call for a catalog and *solid value*

800-421-8006

THE PROGRAMMER'S SHOP™
128- Rockland Street, Hanover, MA 02339
Visa Mass: 800-442-8070 or 617-826-7531 MasterCard

Note: All prices subject to change without notice.

Mention this ad. Some prices are specials.

All formats available.
Ask about POs, COD.

CIRCLE 98 ON READER SERVICE CARD

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing and manipulating C programs. Use C HELPER's UNIX-like utilities which include:

DIFF and **CMP** – for "intelligent" file comparisons.
XREF – cross references variables by function and line.
C Flow Chart – shows what functions call each other.
C Beautifier – make source more regular and readable.
GREP – search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for $135 for MS-DOS, CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: **800-821-2492**

Solution Systems™
335-L Washington Street
Norwell, MA 02061
617-659-1571

CIRCLE 99 ON READER SERVICE CARD

PROLOG-86™

Become Familiar in One Evening

The tutorials combined with the interactive PROLOG-86 Interpreter help you learn the fundamentals of PROLOG quickly. In a few days you should be able to modify sophisticated sample programs included with the product like:

- **an EXPERT SYSTEM**
- **a NATURAL LANGUAGE INTERFACE.**

1 or 2 pages of LOGIC and FACTS create a significant PROLOG program that might take 10 to 15 pages in C.

Prototype quickly in PROLOG. Experiment with applications that might otherwise require a "definitive requirements specification".

Programming experience is not required to use PROLOG-86, but a logical mind is.

PROLOG-86 supports the de facto standard established by "Programming in PROLOG". Ask about the "Artificial Intelligence Concepts" CONTEST. $1,000 prize.
FULL REFUND if not satisfied during first 3 weeks.

Intro Price: $125 for PCDOS, MSDOS or CP/M-86. Most formats available.

For questions/orders, call
800-821-2492
Use Visa, MC, COD.

Solution Systems
335-L Washington St.
Norwell, Mass. 02061
617-659-1571

CIRCLE 97 ON READER SERVICE CARD

EXOTIC LANGUAGE OF THE MONTH CLUB

MUMPS: A multi-user, data base language

By J. Edward Volkstorf Jr.

Imagine an interactive programming language where all data is simply a string of characters. There are no integers vs. real numbers concerns. For instance, the length of the string "BLUE", the number 3.14, and the integer 4321 are all 4.

Conversion from numeric to string never takes place; numbers are already strings! You can use numbers and alphanumeric strings in any arithmetic operation. Of course 2 * 3.14 is 6.28, but "2APPLES" + "3ORANGES" = 5. (Who said you can't mix apples and oranges!) The rule is that a string used as a number is converted to the number until a nonnumeric value is encountered.

Furthermore, even array subscripts are considered character strings. Thus an array such as *X(1)* with a value of 23 is typical, but how about *ACODE("NYC")* having the value of 212? Of course alphabetic and numeric values can co-exist in the same array.

Sounds a lot like SNOBOL, but this is actually a quick start in looking at MUMPS, a unique language that has been around since the early 1970s. It has many rather unusual but very powerful data management features. To those accustomed to FORTRAN, COBOL, or BASIC and the like, MUMPS appears quite odd, plus hard to use and understand. However, this doesn't have to be the case.

It is really quite easy to work with MUMPS. In particular, it has enabled the design of many complex multi-user systems in which interactive users all share in the access to and update of data bases. Before we consider this level of MUMPS, let's look more closely at some of the language's more basic features.

Since all data is represented as a string, there are many string related functions, as in BASIC. For example, to retrieve some portion of a string you use the *EXTRACT* function. Thus

```
$EXTRACT("Computers",I,J)
```

returns the *I*th through *J*th characters from the first argument to the function. If *I* is 4 and *J* is 7 then the function value is "pute".

Note that functions are distinguished by a leading dollar sign (otherwise we would have designated a three dimensional array). Variables may be up to eight alphanumeric characters that begin with a letter. Other string related functions include:

- *$LENGTH(X)* returns the number of characters, i.e., the length of variable *X*
- *$CHARACTER("A")* equal to 65, returning the ASCII code for a character
- *$FIND("COMPUTE","MP")* has the value 3, finding the first occurrence of the string "MP" in the first argument *COMPUTE*
- *$PIECE("1ST,2ND,3RD",",",N)* returns the *N*th comma delimited portion of the first string argument, such that if N is 2 then 2ND is returned. The second argument, the comma here, determines the delimiter

Of course functions can be nested, and numeric values can be used as arguments. For example,

- *$FIND(N,".")* will be either 0 if N is an integer (i.e., no decimal point) or some value greater than 0 if it is a real number
- *$LENGTH($PIECE(NAME,",",1))* might be used to calculate the length of someone's first name, those characters preceding the first comma in the string NAME

As mentioned initially, arrays may also contain strings. They may be any dimension, and subscripts may be ASCII character strings. A typical array might be used, as in most languages, to store a table of numbers:

```
X(1)=12, X(2)=23, X(3)=34, etc.
```

but since MUMPS allows those subscripts to be character strings you can design data tables that look like:

```
ACODE("SAN DIEGO")=619
ACODE("LOS ANGELES")=213
ACODE("SACRAMENTO")=916
```

where the array *ACODE* defines the valid telephone area code numbers for a set of cities. For data such as a person's name, street address, arbitrary words, and the like, this allows some incredible savings in program code when manipulating data. But how do you retrieve the values?

In the first example the array *X* could use a subscript variable that ranges from 1 to 3. MUMPS handles the area code array differently, using the *$NEXT, $ORDER*, and *$DEFINE* functions.

Both *$NEXT* and *$ORDER* take a subscripted variable and determine the subscript value that follows in that array based on the MUMPS collating system. (Note that this gets tricky, since for numeric subscripts 123 follows 45 which follows 6, but for a strict string collating sequence, left to right character comparisons are made; the sequence would be 123, 45, then 6). MUMPS determines whether subscripts are collated as numerics or as ASCII character strings.

Thus for ACODE, the following holds:

- *$NEXT(ACODE("LOS ANGELES"))* would be "SAN DIEGO"
- *$NEXT(ACODE("SA"))* would be "SACRAMENTO"
- *$NEXT(ACODE("SAN DIEGO"))* is −1, indicating that no further subscripted values were found.

The last example is where *$NEXT* and *$ORDER* differ. The latter returns a null string instead of the −1.

There is another way to determine if a single value exists using the *$DATA* function. It takes any variable name as an argument and returns 1 if the variable is defined, a 10 if it is a pointer to a lower level (such as the first subscript in a two dimensional array), or 11, which covers both cases. Otherwise a zero is returned. Thus,

- *$DEFINE(ACODE("SAN DIEGO"))* returns 1
- *$DEFINE(ACODE("SANTA CLARA"))* returns 0
- *$DEFINE(ACODE)* returns 10
- but *$DEFINE(AREAC("LOS ANGELES"))* returns 0 since *AREAC* is not the correct array name (assuming it is not used).

Along with this set of string functions is a host of numeric and string operators. Many are very powerful in nature. For numerics, the typical arithmetic operators of +, −, *, / are used for addition, subtraction, multiplication and division. Also an integer divide is symbolized by \ and modulo operator by #. The last two operators make for very concise expressions such as

```
123 \ 10 = 12
123 # 10 = 3
123 \ 10 # 10 = 2
```

For string operations, the underline designates concatenation. The right bracket is the contains operator which tests whether the first argument contains the second. The left bracket signifies a string follows comparison. As you might expect, the usual Boolean operators, =, <, >, <=, and >=, are used for assorted equality and inequality checks.

Keeping track of operator precedence is simple in MUMPS; there isn't any. Operators are processed in a strict left to right sequence except for expressions in parentheses which, as expected, are processed first.

One of the more powerful Boolean operators is the ?, which tests for pattern matching. A 1 or 0 truth value is returned depending on whether or not the first argument follows the pattern designated by the second argument. Patterns are determined by various character class codes and string constants, which can be modified by a numeric repeat count. For example, in the following statement the letter *N* designates numeric digits that test whether the variable *SSN* is a valid social security number:

```
IF SSN ? 3N1"-"2N1"-"4N
```

Thus the pattern is three digits, a dash, two more digits, a dash and ending with four digits.

In the following example, the letter *U* indicates upper case alphabetic characters and *A*, an alphabetic character. The period is used to mean any number of characters as a repeat count. The expression

```
IF NAME ? 1U.A1",“1U.A
```

might be used to check for a person's name, formatted as a last name and first name separated by a comma.

Other letter codes include *C* for control characters, *E* for any character (everything), and *P* for punctuation.

Logical expressions are combined with & for *and* and *!* for *or*. The apostrophe may be used in front of any logical operator for the inverse of the operator. Thus *IF X < 0 !(X > 1)* tests whether *X* is less than zero or if it is greater than one, and *IF M'[“.” & (M ?.N)* tests whether M is all digits and does not have a decimal point.

Another powerful operator is @, which represents indirect reference of "indirection." Indirection is a unary operation that is performed on character strings. The result of an indirection operation depends on how it is used. Basically, @ takes a value and treats it like a variable:

```
SET NAM="X" ; variable NAM gets
   value "X"
SET @NAM=2 ; indirectly sets vari-
   able X to value 2
```

or an expression:

```
SET ARG="X=3,Y=4" ; define an
   argument
SET @ARG ; sets X to 3 and Y to 4
```

Indirection is useful in processing differently named arrays with one statement.

As in BASIC, input and output in MUMPS is oriented around interactive video display and printer terminals. The command

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

INTROL-C/6809,Version1.5

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident compilers are available under **Uniflex, Flex** and **OS9.**

Cross-compilers are available for **PDP-11/UNIX** and **IBM PC/PC DOS** hosts.

Trademarks:
Introl-C, Introl Corporation
Flex and Uniflex, Technical Systems Consultants
OS9, Microware Systems
PDP-11, Digital Equipment Corp.
UNIX, Bell Laboratories
IBM PC, International Business Machines

For further information, please call or write.

INTROL
CORPORATION
647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

CIRCLE 32 ON READER SERVICE CARD

WRITE outputs the values for constants, variables, and expressions to the currently assigned I/O device.

Input is done in a similar manner using the *READ* command. For example, the code in Listing 1 reads a set of integer values and prints their sum. Note the pattern matching command and the entry of a null string "" to indicate no more data. Comments are denoted in MUMPS as all text that follows a semicolon on a single line.

MUMPS program lines begin with one or more spaces or with an alphanumeric line label. Statements are separated by a single blank space. Since the syntax of each statement determines its format and overall length, statement delimiters are not needed. Control may be passed to named lines with the *GOTO* or *DO* commands.

The *SET* command assigns values to any number of variables. Note that the *IF* command does not use a *THEN*. MUMPS continues processing with the command that follows the logical expression if it is true. Otherwise control is transferred to the beginning of the next line.

An *ELSE* command is available but not in the sense of most other languages. Instead MUMPS has an internal last truth value that may be checked at anytime with the *ELSE*.

Thus an *IF* might have a corresponding *ELSE* located many lines of code away. The *IF* can test this internal truth value by using a null expression, designated by two blank spaces after the *IF*. This means the results of an *IF* evaluation can be checked in other areas of a program without the original expression being coded.

Loops in MUMPS are coded using the *FOR* command. *FOR* repeatedly sets a variable to some value based on one of the four following basic formats:

```
FOR I=1:1:10 ; sets I to 1,2,3 ... 10
FOR N="TOM","DICK","HARRY" ;
    sets N to each of the strings that
    follow
FOR I=1:1 UNTIL I>99 ; increments I
    until the condition following the
    UNTIL is true
FOR I=1:1 WHILE I<100 ;
    increments as long as the condi-
    tion after the WHILE is true
```

The scope of the *FOR* is only to the end of the line. Thus the following counts how many numbers are greater than 0 in the 100 element array *NUM*:

```
SET GZ=0 FOR I=1 TO 100 IF
    NUM(I)>0 SET GZ=GZ+1
```

The *DO* command calls subroutines. It is like the *GOTO* in all respects except that anywhere in the code to which control is transferred, a *QUIT* statement can be

```
     SET TOTAL=0,CNT=0 ; initialize total and count variables
IPN  READ "Enter a number: ",NUMB
     IF NUMB="" GOTO DONE ; exit on a null entry
     IF NUMB'?.N WRITE NUMB," is not valid" ; validate number
     ELSE SET TOTAL=TOTAL+NUMB,CNT=CNT+1 ; update values
     GOTO IPN ; get next number
DONE WRITE "Read ",CNT,"numbers for a total of ",TOTAL
     HALT ; end of program
```

Listing 1.

OASYS
TOOLKIT
SNAP-SHOT

WIZARD
C-8086
CROSS AND NATIVE OPTIMIZING COMPILERS
FOR 8086/87/88/186/286

As part of OASYS' *"ONE STOP SHOPPING"* service for software engineering tools, we are proud to announce the addition of WIZARD C to our integrated collection of more than 50 professional programming tools (e.g. compilers, assemblers, linkers, debuggers, simulators & translators) for M68000, Intel 8086/80186 and NS32000 micros.

WIZARD C benchmarks (against Lattice, Microsoft) prove that it is, by far, the most advanced, full featured, fastest, tightest, optimizing C compiler now available for cross and native (PC) development. Here's why . . .

FEATURES

- Complete K&R implementation plus V7, III extensions
- Supports 8087 Floating Point
- Built-in **LINT**
- Long, medium, short models
- 190+ UNIX III functions — complete run-time library
- In-line assembly allowed
- 100+ extensive warnings/diagnostics
- Intel and Microsoft compatibility at source and object levels
- Written in C — easily ported
- Comprehensive bound documentation
- Supports DOS 2.0, 2.1, IBM/BIOS

SUPPORT TOOLS

- Symbolic C Source Level Debugger (CDEBUG™)
- 100% Intel compatible structured Macro Cross Assembler, Linker/Locater and Librarian
- 8086 Simulator
- Floating point math package (40+ functions)
- C Time Profiler (CLUE™)
- Checkout compiler (SAFE-C™)
- Comm. utilities for up/down loading to MDS, TEK, Microtek

AVAILABILITY

NATIVE: PC/XT/AT using MS/DOS, PC/DOS (Xenix soon)

CROSS: VAX/VMS, Bsd 4.1, 4.2, III, V; 8086's, 68000's (All Unisoft III, V; ports); Callan, Masscomp, Sun, Pyramid, dozens more . . .

Call for pricing, OEM, Site, Corporate, Source and Maintenance licensing information.

OASYS
60 ABERDEEN AVENUE
CAMBRIDGE, MA 02138
(617) 491-4180

W86CH-11/84

CIRCLE 75 ON READER SERVICE CARD

executed causing a return to the next expression in the calling *DO* statement. A *DO* may call one or more subroutines such as:

```
DO SUB1,SUB2,SUB3 ; perform in
     succession the 3 subroutines
```

Related in some respects to indirection is the *XECUTE* command. It evaluates its argument as a command string to be executed as though it were explicitly coded in the program. For example,

```
XECUTE $PIECE("GOTO NEXT/DO
     PREV","/",N)
```

either does a *GOTO NEXT* or a *DO PREV* depending on whether *N* has the value of 1 or 2. Obviously some very cryptic and run-time dependent code can be created with this command.

A noteworthy coding format in MUMPS allows conditional execution of all commands, with minor exceptions. A command name may be followed by a colon and some expression. The expression is evaluated and, if true, the command is processed. Otherwise it is bypassed. Thus the *FOR* loop counting example could be coded

```
SET GZ=0 FOR I=1 TO 100
     SET:NUM(I)>0 GZ=GZ+1
```

A *GOTO* or *DO* command also may use this conditional execution as in:

```
DO:X="YES" SAVE ; call subroutine
     SAVE if X is YES
GOTO:I'=0 SUM ; goto line labeled
     SUM if I is not 0
```

The expression can also follow arguments to the *DO* and *GOTO*, which is handy for selectively executing certain subroutines:

```
DO
     SUB1:X<0,SUB2:X=0,SUB3:X
     >0,SUB4
```

which performs one of the subroutines *SUB1*, *SUB2*, or *SUB3* depending on the value of *X* and always calls *SUB4*.

By now many readers may think MUMPS is a variation of BASIC but without line numbers and numeric and string data typing. To some extent this is true. Where MUMPS truly leaves all other languages behind is in its ability to process complex data structures that are disk resident.

MUMPS files do not exist in the traditional sense. All disk data is a string of ASCII characters, named in the same manner as local memory-resident variables, with the exception of a ^ (carat) preceding the name. The significance of this is that all statements, commands, functions, operators, etc., may use disk variables in a manner identical to local variables.

MUMPS calls its disk data globals since disk data may be shared by all users. Memory-resident variables, on the other hand, are local to one particular MUMPS process. Thus instead of opening, reading, and writing a file, you simply *SET* the global to some value. For instance, the following statement adds one to a status global and then displays its current value:

```
SET ^STAT=^STAT+1 WRITE "Status
     is",^STAT
```

The *$DEFINE* function, mentioned earlier, determines whether a global exists or not. Thus a statement like the one below might precede the above update of global *STAT*

```
IF $DEFINE(^STAT)=0 WRITE "Status
     not defined"
```

These two examples illustrate the simplest forms for using a global. More often globals are processed as multi-dimensional arrays where the subscripts

How Do You Measure A Good C Compiler?

CODE SPEED & SIZE
The Lattice C Compiler "generates code that is quite compact and fast running." *Peter Norton, PC Magazine*

CONSISTENT RELIABILITY
"The Lattice Compiler has performed reliably and predictably." *R. Phraner, Byte Magazine*

COMPILE TIME
"Lattice is a real performer." *Houston, Brodrick, Kent, Byte Magazine*

THIRD-PARTY LIBRARIES
More than 40 library products are currently available for Lattice

UNIX V COMPATIBILITY
The Lattice Library is UNIX V-compatible

DEBUGGER SUPPORT
The Lattice C-SPRITE Debugger is now available

DOCUMENTATION
Lattice "is thorough and excellent." *D. Clapp, PC Magazine*

UPDATE POLICIES
Lattice provides free bugfix updates for 90 days

COOPERATING PRODUCTS
New LMK Utility, dBC Library, CVUE Screen Text Editor, CURSES Screen Library and GSS Graphics are available from Lattice

VENDOR REPUTATION
Lattice is used in more commercial products than any other C Compiler. No run-time license is required.

ALL MEMORY MODELS
Lattice C has 7 memory models available to allow the best solution for the task at hand

AVAILABILITY OF CROSS COMPILERS

SATISFACTION GUARANTEED!

Ask About Our "Trade Up To Lattice C" Policy

LATTICE®

Lattice®, Inc.
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950
TWX 910-291-2190

UNIX is a trademark of AT&T Bell Laboratories

CIRCLE 53 ON READER SERVICE CARD

are either numeric or string subscripts. Processing a global then takes on the form of array processing using the *FOR* loop and *$NEXT* or *$ORDER* functions.

A strict numeric sequence of integers might be used as subscripts to store a series of statistical values, where each value in the global is a single number. For example, let the global ^X contain 200 values. Then their sum is simply:

```
SET TOTAL=0 FOR I=1:1:200
    TOTAL=TOTAL+^X(I)
```

which is identical to how the values would be totalled if *^X* were a local array.

As with local arrays, global subscripts are also defined sparsely, i.e., only those values defined are actually stored in the global. An accounting system could use a global to track the status of purchase orders where the subscript is a purchase order number. The data stored at a particular node in the global might contain items such as the date ordered, the customer number, and so on. For further detail on the data—the purchase order in this example—a second set of subscripts might be used for information about the nature of the order, the line items in it, terms, etc.

We could have a situation where the code presented in Listing 2 would tell a user if a purchase order exists and the date when it was made based on the first subscript in global *^PONUMB*.

A second level of subscripts might be the line item numbers in the *PO*, with a subscript of zero indicating how many lines, plus other related information. When a *PO* is confirmed to exist using the previous example, the line items might be printed, unformatted, as in Listing 3.

The unformatted data is noted since the values in the string would usually be variable length with some special character delimiter. The *$PIECE* function is then used to determine individual field values. A line item node in the above global might look like:

```
12 ^ MX102 ^ DS/DD Diskettes ^ 1.50
    ^ 18.00
```

which could indicate 12 model MX102 DS/DD diskettes at $1.50 each for a total of $18.00. The ^ is an often-used MUMPS field delimiter.

```
PON  READ "Enter PO number: ",PO
     IF PO'?.N WRITE "Invalid number" GOTO PON
     IF $DEFINE(^PONUMB(PO))=0 WRITE "Not on file"
     ELSE WRITE "Ordered: ",^PONUMB(PO)
     GOTO PON
```

Listing 2.

```
SET LN=^PONUMB(PO,0)
WRITE LN," line items for PO ",PO
FOR I=1:1:LN WRITE !,"Line ",I," ",^PONUMB(PO,I)
```

Listing 3.

OASYS
TOOLKIT
SNAP-SHOT

C-68000
CROSS AND NATIVE OPTIMIZING COMPILERS
FOR 68000/10 (and 68020 SOON)

OASYS offers a *"ONE STOP SHOPPING"* service for software developers in need of proven 8-, 16- and 32-bit cross and native tools for Unix and non-Unix 68000, 8086 and 32000 systems. Our critically acclaimed and widely used 68000 tool kit offers high quality, reliable, cost-effective tools.

The OASYS 68000 tool kit consists of Green Hills compilers (C, Pascal and FORTRAN), our own M68000 Macro Assembly Development package, and dozens of other OASYS compatible support tools. Simply stated, we beat the competition on price, speed and tightness of emitted code.

C-68000/10
- Full K & R with Western Electric and Berkeley extensions
- Complete run-time library available as source. No royalty if passed on.
- Supports DEC & IEEE Floating Point
- Integrated optimizer: 30% tighter code than Portable C; 4 times faster
- Generates M.I.T. or EXORmacs assembly source code
- Interfaces to all OASYS tools and Pascal, FORTRAN and PL/M-68K compilers
- Ideal for cross development of boards with no OS, a kernel OS (e.g. VRTX, PSOS, MTOS), or Unix based 68000's

68000/10 Assembly package
- EXORmacs compatibile Macro Assembler, Linker, Librarian, and Cross Reference Utility
- Generates S-records and a.o.
- PIC and reentrant code
- Used 2 years in house
- Over 3,000 sold to date
- Runs on VAX, Prime, PDP-11, 68000's, 8086/88 (PC)
- Written entirely in C

Coming soon
- 68020 C and Cross assembler

Other tools
- Symbolic C Source Debugger
- 68000 Simulator & Disassembler
- C Linecount and Time Profiler Utility (CLUE™)
- LINT for VAX/VMS
- Check Out compiler (SAFE-C™)
- Communications tools

OASYS
60 ABERDEEN AVENUE
CAMBRIDGE, MA 02138
(617) 491-4180

C68K-11/84

CIRCLE 80 ON READER SERVICE CARD

NGS FORTH

A FAST FORTH OPTIMIZED FOR THE IBM PERSONAL COMPUTER AND MSDOS COMPATIBLES.

- *79 STANDARD
- *FIG LOOKALIKE MODE
- *PC-DOS COMPATIBLE
- *ON-LINE CONFIGURABLE
- *ENVIRONMENT SAVE & LOAD
- *MULTI-SEGMENTED
- *EXTENDED ADDRESSING
- *AUTO LOAD SCREEN BOOT
- *LINE AND SCREEN EDITORS
- *DECOMPILER & DEBUGGING AIDS
- *8088 ASSEMBLER
- *BASIC GRAPHICS & SOUND
- *NGS ENHANCEMENTS
- *DETAILED MANUAL
- *INEXPENSIVE UPGRADES
- *NGS USER NEWSLETTER

A COMPLETE FORTH DEVELOPMENT SYSTEM.

PRICE: $70

PLEASE INCLUDE $2 POSTAGE & HANDLING WITH EACH ORDER.

CALIFORNIA RESIDENTS : INCLUDE 6.5% SALES TAX.

NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

CIRCLE 74 ON READER SERVICE CARD

MUMPS provides a syntactic convenience in processing globals, called the naked reference. It essentially means that a global reference may be made without a name and the last level in the global will continue to be processed. This means the global coded *^PONUMB(PO,I)* could have been coded as simply *^(I)* since the initial reference was in the first line at the second level subscripts.

With two or more subscripts used to define their data, globals take on the form of tree structures. Although not a data base management system in the fullest sense, MUMPS provides all the language features needed for creating any hierarchical data base.

Most MUMPS systems implement their global data structures using some form of balanced *B** or multiway tree structure. This guarantees a reasonable response time to examine any node in any size global.

MUMPS is also a multi-user data management system. Its global data structures may be referenced by any user on the system. For instance, the purchase order global could be referenced by any number of users on a system checking the status of their purchases. The typical MUMPS system has all global references occurring in a buffer pool (sometimes called a disk cache) so that each user has access to the same and most current data.

Updating a file is another matter. To control the sequence in which a global is updated there is a *LOCK* command. This allows a user to have exclusive control of an entry in a lock table that in turn typically defines an area of a global.

In the purchase order example, a receiving area might be changing the status of some line items in the file. In the clerical area a buyer might also change something in the *PO* that is on back-order. Either user could have their program *LOCK* the entire order if both used a statement of the form:

```
LOCK ^PONUMB(PO)
```

On the other hand, if the changes are only specific to one line item in the order then the control may be:

```
LOCK ^PONUMB(PO,LINE)
```

where *PO* is the purchase number and *LINE* the line number being changed.

Users on the system making inquiries do not have to use the locking code since they will always be retrieving the most current *PO* data from the buffer pool.

In general, access codes, passwords, and the like on commercial MUMPS systems provide an overall control mechanism. They usually map a group of related globals to a certain user class when those users sign on to the system. Many variations exist, even to the extent of coding certain globals as *read only*, or *read/write* by user class.

The *HANG* and *JOB* commands are also useful in a multi-user environment. The *HANG* command suspends program execution for a specified number of seconds. This might be used in time dependent processes, such as waiting until a certain time of day has arrived before continuing with program execution.

The *JOB* command initiates another MUMPS process. This might be something as simple as a print spooler to a process that files the records generated by a number of other data entry processes. With the *HANG* command the second process could be set to function every few minutes or so instead of continually waiting for records to file.

At this point I would like to describe the typical MUMPS system layout. A single-user system consists of basically the interpreter and the single-user area. The user area has a fixed space that describes the current context in which the user is functioning. Three variably sized areas also exist: the user's program code, the local variables (the symbol table), and a stack area for intermediate values during program execution. (Figure 1.)

For multi-user systems, an executive handles the time sharing and context switching between users. In the disk buffer pool all global references are handled. (Figure 2.)

This area is also used to hold program routines called in from the disk by the *DO* and *GOTO* commands. Some venders even share program code between users. Thus the user partition only consists of the fixed context area, the symbol table, and stack.

Some systems do not fluctuate just in a fixed partition area and thus allow the symbol area to be completely dynamic in size. Some systems even swap partition overflows to and from the disk. In these, MUMPS becomes a virtual machine much in the same manner as the IBM 360/370 OS.

Some MUMPS vendors who offer networked systems provide another very sophisticated area of global processing. In these systems a global has some additional coding within its name designating that it resides on another machine. For example, some vendors use brackets to indicate the networked machine as in

```
SET
    ^[MACH3]GLOBAL(SUB1,SUB2)
    =0
```

UNIX SOFTWARE

UniPress Product UPDATE

AMSTERDAM COMPILER KIT FOR UNIX™

- Package of C and Pascal compilers, cross compilers and assemblers. The Kit was developed at the Vrije University of Amsterdam and has been used in Europe for years.
- Available for a wide variety of host and target machines: Unix 68000, VAX and PDP-11 and can produce code for any of those machines, plus the 8086.
- Collection of programs designed to simplify the task of producing portable compilers and interpreters.
- The Kit contains complete internals documentation describing how to make modifications needed to add a new program language or a new target machine.
- Cross assemblers are also provided for 8080, Z80, Z8000, 8086, 6800, 6809, 68000, 6502, and PDP-11. Cross interpreters, for testing, are also included.
- The Kit consists of 8 components: preprocessor, front end, peephole optimizer, global optimizer, back end, target machine optimizer, universal assembler, utility package.
- The Kit includes all sources and documentation for VAX/Unix 4.1/4.2, PDP-11/V7, and MC68000 implementations.

Source

Price: Full system $9950
Educational Institution $995
Source requires an AT&T source compiler license

Binary

Full system $4995
(contact us for specific availability)

OEM terms available • Much more Unix software, too! • Call or write for more information.

UniPress Software, Inc.

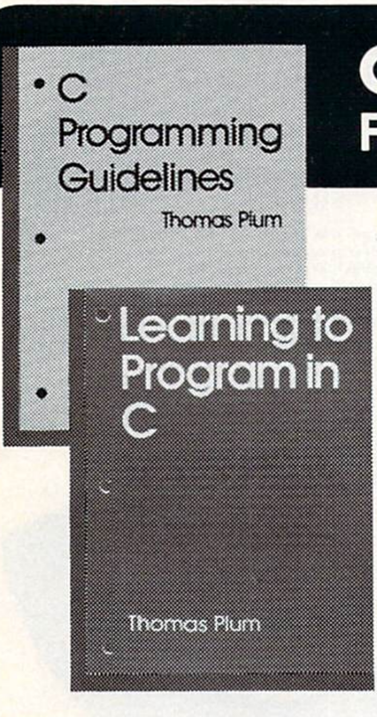
2025 Lincoln Highway, Edison, NJ 08817
201-985-8000 • Order Desk: 800-222-0550 (outside NJ)
Telex 709418 • Mastercard and Visa

Lattice is a registered trademark of Lattice, Inc. Unix is a trademark of Bell Laboratories. MS-DOS is a trademark of Microsoft.

CIRCLE 38 ON READER SERVICE CARD

C LANGUAGE PROGRAMMING

From Plum Hall...the experts in C training



Learning to Program in C 372 pp., 7½" x 10", Price $25.00
A practical, step-by-step guide for everyone acquainted with computers who wants to master this powerful "implementer's language" Inside, you will learn how to write portable programs for the full spectrum of processors, micro, mini and mainframe

C Programming Guidelines 140 pp., 7½" x 10", Price $25.00
A compilation of standards for consistant style and usage of C language. Arranged in manual page format for easy reference, it presents time-tested rules for program readability and portability.

FREE
C LANGUAGE POCKET GUIDE!

A handy C language programming pocket guide is yours *free* when you order either (or both) of the manuals above. A full 14 pages of valuable C language information!

PLUM HALL
1 Spruce Av, Cardiff NJ 08232

The experts in C and UNIX™ training.
Phone orders: 609-927-3770

Please send me: ______ information on C and UNIX Training Seminars
______ copies of Learning to Program in C @ $25.00/copy
______ copies of C Programming Guidelines @ $25.00/copy

NJ residents add 6% sales tax.

NAME ______
COMPANY ______
ADDRESS ______
CITY/STATE/ZIP ______
☐ Check ☐ American Express ☐ Master Card ☐ Visa
CARD # ______ EXP. DATE ___/___ Signature ______

UNIX is a trademark of AT&T Bell Laboratories

CIRCLE 50 ON READER SERVICE CARD

WHY WOULD ANY SANE PERSON SPEND $199 FOR A BetterBASIC SYSTEM WHEN DOS's IS FREE?

HERE ARE 10 REASONS: TEST YOUR SANITY

1. Full support for 640K memory **2.** Structured language with BASIC syntax **3.** Separately compiled program modules **4.** Speed: FAST **5.** Extensibility (Make your own BASIC.) **6.** User-defined procedures and functions **7.** Built-in windows support **8.** Interactive programming language based on an incremental compiler **9.** 8087 math support **10.** Runs on IBM PC, IBM PC/XT and compatibles

Summit Software
Technology, Inc.™
P.O. Box 99, Babson Park
Wellesley, MA 02157

(617) 235-0729

NOW AVAILABLE FOR THE TANDY 2000 & 1200

Better BASIC™

Sane Programmers Order BetterBASIC Now
Price: $199 8087 Math Module: $99
Runtime System: $250 Sample Disk: $10

BetterBASIC is a trademark of Summit Software Technology, Inc. IBM PC, IBM PC/XT and PC/DOS are trademarks of International Business Machines Corp. MS-DOS is a trademark of Microsoft Corp.

MasterCard, VISA, P.O. Checks, Money Orders, and C.O.D. accepted.

CIRCLE 88 ON READER SERVICE CARD

8087 SALE

$130 while quantity lasts.

Put the speed and power of the 8087 to work for you.

We've got the Intel ceramic 8087-3 chip which operates at 5 MHz.

Order by phone and we ship the same day for Visa or MasterCard customers. Or you can send cash, check or M.O. (Sorry, no COD's). Add California state tax, if applicable and $3 for shipping in the U.S.A. or Canada; $15 for foreign air mail. **(415) 827-4321**
VISA MASTERCARD

Steve Rank, Inc.
1260 Monument Blvd., Concord, CA 94518

CIRCLE 87 ON READER SERVICE CARD

YOU CAN C

VIEW v3
The ultimate CP/M disk utility.

- **Edits** any disk sector
- **Displays** allocation bit map
- **Rebuilds** files automatically from selected blocks

If you ever had to recover a crashed disk, you need **VIEW**. Use **VIEW** to patch damaged sectors, un-erase files, examine other disk formats, more.
COMPLETE C SOURCE CODE $59

BD Software's C Compiler
The most popular CP/M C Compiler comes with **symbolic debugger** and special Western Wares' **ISIS-II** function library. **$140**

C-PACK
A disk full of useful CP/M utilities written in C, PEPE, DEL, BACKUP, DPATCH, ECHO, CHX, PAUSE, more. Source incl **$19**

Other C Products
EZZAP, ICX, MEDIT, more
Send for a catalog today

CP/M®, Digital Research
ISIS-II ®, Intel Corp

Western Wares **303-327-4898**
Box C • Norwood, CO 81423

CIRCLE 90 ON READER SERVICE CARD

A MUMPS program that *SET*s the global value is actually passing data through a high-speed communications device to another MUMPS computer's disk. Once again, other than the slight name change, the remote global is processed in the same manner as one on the current machine, which is the same as locally defined variable data.

As you can see now, MUMPS is not just a language but a unique way of developing multi-user data base oriented systems on mini and microcomputers. When I think of MUMPS I think of the fanaticism of C or Forth. Like these languages, MUMPS is fostered by a dedicated set of programmers who often work with systems containing over 100 interactive terminals that are updating and accessing the same data base on moderately priced minicomputers.

MUMPS has its roots in the health care environment; in 1969, the initial version of the language was created at Massachusetts General Hospital. The following seven years saw a small proliferation of various dialects which culminated in an ANSI standardized version that was first offered by a number of hardware manufacturers, most notably Digital Equipment Corp. on its PDP-11.

The language is currently available on a wide range of minis, including the Data General Eclipses, Prime, Tandem, and others. MUMPS is also available in a multi-user environment on the IBM PC and 6809-based microprocessors. Single-user systems are also available for the Apple, 6809, IBM PC, and 8080 systems.

MUMPS has an enormous number of applications in the health care field, particularly in laboratory and radiology data management where there is extensive variable length textual data.

Another important application includes overall patient data management for admissions, business office and medical records. Many clinic and physician group practice systems also use MUMPS and of course many business-oriented applications are written in it.

The key element MUMPS offers is its ease in developing multi-user data management systems. In the health care application examples noted, it is essential to allow all users access to individual patient data. For example, in a laboratory application, test results can be viewed as soon as they are entered into the system on video display terminals at the nursing station, physicians' lounge, or even the physician's office if he or she has a modem for dialing into the hospital's computer.

For further information contact the MUMPS Users Group. They offer a wide assortment of publications and services for the novice and veteran MUMPS user. A national meeting takes place every year in June in different parts of the country. For more specific information on actual commercial MUMPS systems, the MUG provides a list of current vendors of MUMPS language processors and MUMPS applications. Four microprocessor-based MUMPS system vendors are also included in the references.

References

1. MUMPS User's Group, 4321 Hartwick Rd., Suite 308, College Park, Md. 20740
2. Micronetics Design Corp., 932 Hungerford Dr., #11, Rockville, Md. 20850, (301) 424-4870. Motorola 6809, 68000, IBM PC MUMPS implementations.
3. Richard Walters, Ph.D., Dept. of Computer Science, Univ. of California, Davis, Calif. 95616. Public domain 8080, 8088, Z80 MUMPS.
4. Innovative Computer Systems Inc., 1660 S. Albion, Ste. 902, Denver, Colo. 80222, (303) 759-7633. IBM PC implementation.
5. Eclectic Systems Corp., 16260 Midway Rd., Addison, Texas 75001, (214) 661-1370. Various microprocessor implementations of MUMPS.

J. Edward Volkstorf Jr. is a free-lance software consultant living in Chesapeake, Va.

```
Single user MUMPS system memory layout

 -------------
|             |
|             |  Interpreter
|             |
|             |
 -------------
|             |  Fixed area
 -------------
|             |  Program area
|             |
|             |  Stack
|             |
|             |  Symbol table
 -------------
```

Figure 1.

```
Multi-user MUMPS system memory organization

 -------------
|             |  Time sharing executive
 -------------
|             |
|             |  Interpretor
|             |
|             |
 -------------
|             |
|             |  Disk cache/buffer pool
|             |
 -------------
|             |
                 User
|             |
|             |  partition
|             |
|             |  areas
|             |
 -------------
```

Figure 2.

WIZARD C

Fast compiles, fast code and great diagnostics make Wizard C unbeatable on MSDOS. Discover the powers of Wizard C:

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

The new standard for C Compilers on MSDOS!

Only $450

WSS

For more information call (617) 641-2379
Wizard Systems Software, Inc.
11 Willow Ct., Arlington, MA 02174
Visa/Mastercard accepted

CIRCLE 86 ON READER SERVICE CARD

RP/M T.M.

By the author of Hayden's "CP/M Revealed."

New resident console processor RCP and new resident disk operating system RDOS replace CCP and BDOS without TPA size change.

User 0 files common to all users; user number visible in system prompt; file first extent size and user assignment displayed by DIR; cross-drive command file search; paged TYPE display with selectable page size. SUBMIT runs on any drive with multiple command files conditionally invoked by CALL. Automatic disk flaw processing isolates unuseable sectors. For high capacity disk systems RDOS can provide instantaneous directory access and delete redundant nondismountable disk logins. RPMPIP utility copies files, optionally prompts for confirmation during copy-all, compares files, archives large files to multiple floppy disks. RPMGEN and GETRPM self-install RP/M on any computer currently running CP/M®2.2. Source program assembly listings of RCP and RDOS appear in the RP/M user's manual.

RP/M manual with RPMGEN.COM and GETRPM.COM plus our RPMPIP.COM and other RP/M utilities on 8" SSSD $75. Shipping $5 ($10 nonUS). MC, VISA.

118 SW First St. - Box G
Warrenton, OR. 97146

Micro Methods, Inc.
(503) 861-1765

CIRCLE 52 ON READER SERVICE CARD

PRESENTING THE MEGAMAX C COMPILER

FEATURING:
• IN-LINE ASSEMBLY • ONE PASS COMPILATION • SUPPORT OF DYNAMIC OVERLAYS • FULL ACCESS OF MACINTOSH TOOLBOX ROUTINES • AND MUCH MORE ...

DEVELOPMENT SYSTEM PACKAGE INCLUDES:
• FULL-SCALE IMPLEMENTATION (K&R) C COMPILER • THE STANDARD C LIBRARY • ROM ROUTINES LIBRARY • LINKER • LIBRARIAN AND DOCUMENTATION...

DEALER AND USER GROUP INQUIRIES INVITED

NEW FOR THE MACINTOSH

$299.95

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:
Megamax, Inc.
BOX 851521, DEPT. 1
RICHARDSON, TX 75085-1521
(214) 987-4931

MACINTOSH IS A REGISTERED TRADEMARK OF APPLE COMPUTER INC.

CIRCLE 13 ON READER SERVICE CARD

YOU DON'T NEED
EUREKA!™ ??
CONGRATULATIONS!!

We admire your talents. After all, few people can remember where to find that six month old letter to Wonder Waffle Works, or which of the twenty versions of IMPORTNT.BAS is the one you need yesterday.

Or maybe we should envy your spare time. Ah, to be able to haul out a stack of disks, slip each one into a drive, browse through the directory, and TYPE the various prospects to find that one file or program.

Or perhaps you're the adventurous type who thrills to the challenge of groping through scantily labeled disks, cheering that magical moment when hidden treasures are uncovered.

On the other hand, it occurs to us that you just may not know the advantages of **EUREKA!**, the fast, menu driven disk cataloger for CP/M. **EUREKA!** puts your entire disk library at your fingertips. Files may be found quickly and easily - by name or by comments you can put in the file itself. Of course the manual includes a tutorial to help you get started.

Still only $50. Ask your dealer, or contact:

MENDOCINO SOFTWARE COMPANY, INC.

Dept. L-1
P.O. Box 1564
Willits, CA 95490
Phone: (707) 459-9130

I found it!

Add $2.50 Shipping:
Calif. residents add $3.00 sales tax.

We accept VISA
& Mastercharge

CP/M is a registered trademark of Digital Research Corp.
EUREKA! is a trademark of Mendocino Software Co., Inc.

CIRCLE 48 ON READER SERVICE CARD

PRODUCT BINGO

By Doug Millison

Each month Product Bingo features the latest in new software and hardware products of interest to COMPUTER LANGUAGE *readers. Product Bingo items are based on information received from the manufacturer and are not meant to be product evaluations, reviews or endorsements. To find out more about a particular product simply circle the appropriate number on the Reader Service card—you'll receive information directly from the manufacturer.*

Note to manufacturers: Send new product information to Doug Millison, Product Bingo, COMPUTER LANGUAGE, *131 Townsend St., San Francisco, Calif. 94107.*

Application development system

Data Language Corp. has released Progress, a high-performance application development system. In use now on AT&T, Fortune Systems, and Convergent Technologies machines, Progress will soon be available for the IBM PC AT under MS-DOS and Xenix.

Progress combines a powerful data base management system, application language, and an advanced user interface. Automatic screen and report generation, error recovery and an on-line tutorial are featured.

Prices start at $1,450 for single users and $1,950 for multi-user systems. Query/run-time and plain run-time systems are available for sale with applications. A Progress Introductory System is available for $295, including on-line tutorial, full documentation, and all Progress facilities for building a working application limited only by data base size.

CIRCLE 110 ON READER SERVICE CARD

An APL for the IBM PC

From Portable Software comes PortaAPL, a complete implementation of the standard APL language for the IBM PC and other systems.

Written in C for portability, PortaAPL is extended with a full-screen editor, ASCII character set option, Host File System Option, and access to machine language functions. For faster operations, PortaAPL can work with the 8087 math coprocessor on the IBM PC.

In addition to the IBM PC version ($1,945), PortaAPL can be used on the Motorola 68000 ($395), and DEC VAX ($2,995). Source and OEM licenses are available.

CIRCLE 111 ON READER SERVICE CARD

Now your computer talks back

When the going gets tough, the tough start talking. To themselves. Now your computer can make it a dialogue with RACTER, from John D. Owens & Associates. The author of the first book written by a computer program—*The policeman's beard is half constructed*—RACTER boasts the ability to produce an original conversation on a variety of topics, with an eccentric personality and sense of humor. Available now for the IBM PC and CP/M, (coming soon for Apple and Commodore), RACTER retails for $69.95.

If you want to write interactive English, check out INRAC, a prose-creating, high-level programming language. Commands for parsing input and the ability to call subroutines at random allow INRAC to be used to generate original prose, avoid stereotypical responses, parody a given literary style, or simulate specialization in a given subject. The price for INRAC had not been set at press time.

CIRCLE 113 ON READER SERVICE CARD

New S-100 master, slave SBCs

Feature flexibility is the key to Teletek's new SBC 86/87 S-100 slave SBC. Options include 128K to 512K RAM, Intel 8086 CPU at 5MHz or 8MHz, and a 8087 math coprocessor.

The TurboDOS operating system provides CP/M 86 compatibility, while TurboDOS 1.4 provides a PC-DOS (MS-DOS) emulator in a multi-user environment. The SBC 86/87 sells for $837 in OEM quantities.

Systemaster II is Teletek's new S-100 master SBC. Simultaneous control of up to four floppy disks is possible.

Two 64K banks of parity checked RAM provide a TPA of over 63K under TurboDOS 1.3 and 60K under CP/M 3.0. Two RS-232 serial ports provide synchronous or asynchronous communications with software programmable baud rates. Systemaster II sells for $679.50 in OEM quantities.

CIRCLE 115 ON READER SERVICE CARD

Customize screen input programs

Programmers using IBM PC, AT, XT, PCjr and true compatibles can generate bug-free customized screen input programs quickly with Software Bottling Co.'s Screen Sculptor.

The source code generated can be modified or merged with other programs. IBM BASIC, IBM Pascal, and Turbo Pascal capabilities are all included in the package. Screen Sculptor sells for $125.

CIRCLE 121 ON READER SERVICE CARD

Mac Switch makes sharing easy

The people who brought you Mac Inker are at it again. Mac Switch, from Computer Friends, looks like another indispensable device you'll wish you had dreamed up yourself.

Mac Switch is a manual switch used to connect two peripherals to one computer or two computers to the same peripheral. It is available in two versions: parallel centronics interface for printers and serial RS-232 for data communications. Mac Switch retails for $99.00.

CIRCLE 122 ON READER SERVICE CARD

Six Times Faster!

Super Fast Z80 Assembly Language Development Package

Z80ASM

- Complete Zilog Mnemonic set
- Full Macro facility
- Plain English error messages
- One or two pass operation
- Over 6000 lines/minute
- Supports nested INCLUDE files
- Allows external bytes, words, and expressions (EXT1 * EXT2)
- Labels significant to 16 characters even on externals (SLR Format Only)
- Integral cross-reference
- Upper/lower case optionally significant
- Conditional assembly
- Assemble code for execution at another address (PHASE & DEPHASE)
- Generates COM, HEX, or REL files
- COM files may start at other than 100H
- REL files may be in Microsoft format or SLR format
- Separate PROG, DATA & COMMON address spaces
- Accepts symbol definitions from the console
- Flexible listing facility includes TIME and DATE in listing (CP/M Plus Only)

SLRNK

- Links any combination of SLR format and Microsoft format REL files
- One or two pass operation allows output files up to 64K
- Generates HEX or COM files
- User may specify PROG, DATA, and COMMON loading addresses
- COM may start at other than 100H
- HEX files do not fill empty address space.
- Generate inter-module cross-reference and load map
- Save symbol table to disk in REL format for use in overlay generation
- Declare entry points from console
- The FASTEST Microsoft Compatible Linker available

SPEED! SPEED! SPEED!

- Complete Package Includes: Z80ASM, SLRNK, SLRIB - Librarian and Manual for just $199.99. Manual only, $30.
- Most formats available for Z80 CP/M, CDOS, & TURBODOS
- Terms: add $3 shipping US, others $7. PA add 6% sales tax

For more information or to order, call:

1-800-833-3061

In PA, (412) 282-0864

Or write: SLR SYSTEMS

1622 North Main Street, Butler, Pennsylvania 16001

CIRCLE 59 ON READER SERVICE CARD

SLR Systems

8087 AND 80287 TECHNICAL TOOLS

87FFT™ performs Forward and Inverse FFTs on real and complex arrays which occupy up to 512K bytes of RAM. Also does convolutions, auto correlations, hamming, complex vector multiplication, and complex to radial conversions. Callable from MS Fortran or 87BASIC/INLINE..... $150

87FFT-2™ performs two-dimensional FFTs. Ideal for image processing. Requires 87FFT...$75

MATRIXPAK™ manages a **MEGABYTE!** Written in assembly language, our runtime package accurately manipulates large matrices at very fast speeds. Includes matrix inversion and the solution of simultaneous linear equations. Callable from MS Fortran 3.2, 87MACRO, 87BASIC/INLINE, and RTOS......... each $150

DATA ACQUISITION PACKAGE Interactive, user-oriented language which allows the acquisition and analysis of large data streams.......... CALL

GRAPHICS PACKAGES

Energraphics (stand alone)	295
Grafmatic for MS Fortran or Pascal	125
Plotmatic for Grafmatic	125
Halo for Basic, C or Fortran	each 150

OTHER TOOLS

Alpha Software ESP	595
Borland Sidekick, Toolbox, or Graphics	45
COSMOS Revelation	850
Lattice C	299
PSI MATHPAK	75
smARTWORK	895
SPSS/PC	695
STSC APL★PLUS/PC	475

RTOS - REAL TIME OPERATING SYSTEM RTOS is a multi-user, multi-tasking real time operating system. It includes a configured version of Intel's iRMX-86, LINK-86, LOC-86, LIB-86, OH-86, and MicroWay's 87DEBUG. Runs on the IBM-PC, XT, PC-AT and COMPAQ.......... 400

INTEL COMPILERS[1]

FORTRAN-86	750
PASCAL-86	750
PL/M-86	500
87C (LATTICE/MICROWAY)	750
ASM-86	200

URS™ - Universal Run Time System[1] Generates programs with the Intel compilers which run on other operating systems. MS-DOS version is included with RTOS. Xenix-286 Version.......... 300

SoftScope Symbolic Debugger[1] 500

[1] Requires RTOS or iRMX-86. All Intel compiler names and iRMX-86 TM Intel Corp.

87BASIC/INLINE™ converts the output of the IBM Basic Compiler into optimized 8087 inline code which executes up to seven times faster than 87BASIC. Supports separately compiled inline subroutines which are located in their own segments and can contain up to 64K bytes of code. This allows programs greater than 128K! Requires the IBM Basic Compiler and Macro Assembler. Includes 87BASIC.......... $200

PC AT and 86-310 DRIVES

30 MEGABYTE WINCHESTER	2000
53 MEGABYTE WINCHESTER	2600
SYQUEST FIVE MEGABYTE	950

HARDWARE AND LANGUAGES

8087-3 5mhz $149
Including DIAGNOSTICS and 180-day warranty For IBM PC and compatibles

8087-2 8mhz $275
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5mhz $275
For the IBM PC AT

64K RAM Set $30

256K RAM Set $195

128K RAM Set PC AT. $225

NUMBER SMASHER™...CALL
10mhz 8087 coprocessor board for the IBM PC

FORTRAN and UTILITIES

Microsoft Fortran 3.2	239
IBM Professional Fortran	595
Intel Fortran-86[1]	750
FORLIB+	65
STRINGS and THINGS	65

BASIC and UTILITIES

IBM Basic Compiler	270
87BASIC/INLINE	200
Summit BetterBASIC™	175
Summit 8087 Module	87

MACRO ASSEMBLERS

IBM Assembler with Librarian	155
87MACRO	150

PASCAL

Microsoft Pascal 3.2	209
Borland Turbo	45
Turbo with 8087 Support	85

Micro Way

P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

You Can Talk To Us!

NO CHARGE FOR CREDIT CARDS
ALL ITEMS IN STOCK
CALL FOR COMPLETE CATALOG

CIRCLE 57 ON READER SERVICE CARD

SOFTWARE REVIEWS

Expert team analyzes 21 C compilers

By Steve Leibson, Fred Pfahler, Jim Reed, and Jim Kyle

In the early 1970s at Bell Laboratories, Dennis Ritchie developed his C programming language for PDP-11 minicomputers. He needed a systems programming language for the new UNIX operating system created by colleague Ken Thompson. The C language has grown in popularity since those early days due to its simplicity, elegance, and power.

Today C has left the UNIX environment and comfortably resides on other operating systems and computers. This special review will look at the C compilers currently available on the market from a variety of operating system perspectives. We will cover 12 C compilers and one interpreter running under PC-DOS, six CP/M C compilers, two OS-9 C compilers, one CP/M 86 C compiler, and one TRS-80 compiler. (C compilers for the Apple Macintosh will be reviewed in the April issue.) A list of company names, addresses, and phone numbers is included at the end of the review.

Languages have been grouped by operating system in the text and in four tables on bundled software and essential information (page 75), benchmark results (page 79), basic data types (page 99), and library functions (page 100). It may not be fair to compare language implementations across operating system boundaries, but such comparisons are inevitable and interesting.

We tested the C languages by installing them in appropriate systems. Then we attempted to compile and run a suite of four benchmark programs on each. Details on the benchmarks we used—Sieve of Eratosthenes, Fib, Deref, and Matrix—are contained in a sidebar (page 82).

Throughout this review we refer to the definitive reference on C—*The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie. C users all over refer to this book simply as K&R and we will too.

MS-DOS/PC-DOS C COMPILERS

Little doubt exists that MS-DOS, in the guise of PC-DOS in the IBM PC family and the many compatibles, is the successor to CP/M as the most popular microcomputer operating system. With the addition of tree-structured directories introduced with DOS 2.0, this operating system clearly targets low-end UNIX applications.

In this atmosphere, many software houses have decided to offer C compilers for MS-DOS and PC-DOS. As a bonus, one company has introduced a C interpreter! We will look at 11 of these compilers and the interpreter in this portion of the review. All compilers were tested using DOS 3.0 on an IBM PC with 544K of memory and two 360K floppy drives.

The 8088 processor, with its segmented architecture, presents an interesting problem for language developers. Pointers can be either 16 bits or 20 bits long. This has led to the use of four programming models, originally set forth in Intel's PL/M 86 programming language. These models are called small, medium, compact, and large (or big).

The small 8088 memory model defines separate code and data areas, with each area limited to 64K bytes. Thus the total program with data included may use a maximum of 128K bytes of memory. The program stack must also fit in the 64K allocated to the data. Programs compiled with the small memory model use 16-bit pointers.

The medium memory model limits the data segment to 64K bytes, so data pointers within the program are 16 bits long. The program, however, is no longer constrained to a 64K block but can extend throughout memory. Call and return pointers are therefore managed by the compiler and linker so that their size is appropriate for each instance.

Programs compiled under the compact memory model have separate 64K spaces for code, data, and stack segments, while a fourth segment, called the heap, can occupy the rest of memory.

The large or big model allows code to expand throughout memory, while the global data and stack are limited to 64K segments. The heap can also extend throughout memory, as in the compact model.

Lattice and Microsoft have a slightly different set of memory models; the small, program, data and large models allow for a simpler allocation of pointer size. Small programs have a 64K code and a 64K data segment. Programs compiled with the program model have a 1MB program space but a 64K data space, while programs compiled with the data model are limited to a 64K program space and have a 1MB data space. The large model allows for both segments to use 1MB.

Manx's Aztec C

Let's start off with a fine example of a compiler that has many strengths—one from Manx Software Systems with the unlikely name of Aztec C. Two versions of the package are supplied, version 1.06D and 2.20B. The earlier version is slower but more mature (less bugs), and the new version is faster but has not yet aged. We tried both versions, and the results are in the benchmark table (Table 2). Programs compiled with the version 1.06 compiler won't link with the version 2.20 linker and vice versa. You get a "not an object file" error.

Two floating point libraries are supplied. One library does not use the 8087 and the other does. Aztec C supports only the small memory model, but it also supports overlays, so programs can become quite large. The version 2.20 compiler has an option to support either the 8086/8088 or the 80186/80286. We did not quite understand this because the 80186 is more like the 8086 than the 80286, however, we did not have the hardware to test this option.

No batch files were supplied on the distribution disks and we found no recommended files in the manual, but the syntax for the compile and link commands is fairly simple. With the compile disk on drive A and the source code on drive B, *CC B: <filename>* compiles the program and *LN B: <filename> C.LIB* links it.

The Sieve and Fib programs compiled, linked, and ran for both versions of Aztec C, but we had problems with Deref and

Of course, POWER!™ saves your Bad Disk.

NOW! WINDOWS FOR IBM!

It also does 54 other things to keep your disk in line.

EVERYTHING YOU ALWAYS WANTED TO DO, BUT WERE AFRAID TO TRY

Unlike some utility programs that are a headache to use, POWER! is engineered to spoil you with 55 features, simple and uniform commands, and utter simplicity of use. POWER! automatically alphabetizes and numbers your files. You select by the number and never type file names again. Need to **[COPY]**, **[RENAME]**, **[ERASE]**, or **[RUN]** programs? Just type in their menu number! POWER! also locks out your disk's bad sectors **[TEST]** without destroying files–a critical difference from other utilities that search and destroy, without informing you what they've done, leaving you to wonder why your programs won't run. (And POWER! still has 50 commands to go!)

POWER! ONE PROGRAM DOES IT ALL!

You may own a few utility programs for your computer housekeeping, each with its own commands to memorize. POWER! has all the programs rolled into one 16K integrated package, so you do things you've never tried before–every day. Save sensitive data from prying eyes with **[PASS]** word protect, move a block of memory **[MOVE]**, look for data **[SEARCH]** or compare files **[CHECK]**. POWER! also makes easy work of patching, **[DISPLAY/SUBSTITUTE]**, customizing software **[LOAD/SAVE]**. Among the other commands are **[SIZE]**, **[STAT]** **[LOG]**, **[DUMP]**, **[TYPE]**, **[JUMP]**, **[FILL]**, **[SET]**, and the CP/M version lets you restore erased files–even when you don't remember the filename–at a flick of the POWER! **[RECLAIM]** command. (Still 31 commands to go!)

POWER! NOW FOR IBM's PC-DOS AS WELL AS CP/M

We first developed POWER! for CP/M two years ago, and a stack of testimonials from FORD to XEROX testify to its excellence. For IBM-PC™ users, special features like managing sub-directories, **[CHANGE]**, and a separate creation of up to 8 simultaneous, on-screen **[WINDOWS]** have been added.

MONEY-BACK GUARANTEE AND A 10 DAY TRIAL

POWER! has the Seal of Approval from the Professional Software Programmers Association, and you, too, must be happy with POWER!–or your money back! For only $169 you can now really be in control of your computer. Call Computing! at (415) 567-1634, or your local dealer. For IBM-PC or any CP/M machine. Please specify disk format.

The company that earns its exclamation point.

COMPUTING!

2519M Greenwich, San Francisco, CA 94123

TO ORDER CALL 800 TOLLFREE

800-428-7825 Extension 96M

In CA: 800-428-7824 Extension 96M

IBM and IBM-PC are registered trademarks of International Business Machines Corporation.

CIRCLE 37 ON READER SERVICE CARD

Matrix. Deref had to be reduced to 12 levels of indirection before Aztec C could compile it. At that level, the program compiled and linked in 60 sec, ran in 6 sec, and took 4,352 bytes.

Our problem with Matrix turned out to be library related. A second math library, called m.lib, must also be linked with Matrix. After that, we had no further trouble.

Manx offers Aztec in several different flavors, so you can port your programs across a lot of hardware if you need to. One interesting aspect of the Aztec C compiler is the PRO extension. This includes an editor (called z), a symbolic debugger, and source files for several of the library functions. The PRO version of the compiler can generate ROMable code. These extensions seem to be quite valuable for serious development work. The Aztec C86 compiler for MS-DOS or CP/M 86 is $249.00, and the compiler with the PRO options is $499.00.

Supersoft C

Supersoft C went together a little differently from the other packages. It also would not fit on one 360K disk so we were instructed to put just the library on a second disk. Then a single batch file called CM.BAT ran the show.

When it came time to link in the library, the linker paused and asked where the library could be found since it wasn't where it was supposed to be. This was our cue to swap the library into drive A and type "A". Well, that didn't work but typing "A:" did. Another company that doesn't seem to read its own manual.

At $350, Supersoft C seems a bit expensive for a compiler that does not yet support floating point. Supersoft says it plans to upgrade the compiler to floating point but for now it supports Binary Coded Decimal floating point through float and double declarations, plus some special library math routines. It seems that although you can declare floats and doubles, all you can do with them is pass their pointers around. You can't use the compiler's intrinsic operators on them.

Only one memory model seems to be supported by Supersoft C, and it appears to be the small model. Supersoft C does not directly support overlays, but a section in the manual states that Digital Research's RASM86 assembler running under MS-DOS will support overlay operation. Supersoft's linker ELINK supports overlays in the CP/M 80 environment.

In the benchmarks, Sieve compiled and ran fine, we had to remove *void* from the Fib program, Deref ran fine, and Matrix could not be compiled due to the lack of intrinsic floating point math.

Supersoft C seems to be a software package in transition. Until it supports full C, we think it is overpriced. When floating point math is added, it will be a real contender.

Computer Innovations' C86

We liked Computer Innovations' C86 right away. You have to admire a company that names its librarian Marion. However, this fondness quickly dimmed. Executable files in C86 are supplied squeezed to save room on the disk and must be unsqueezed to create the working disk. We don't know about you, but we like to get the software out of the box and get it running fast. Unsqueezing files is for bulletin boards!

Learning how to use C86 isn't as easy as it might be. No test program is supplied, although there is a listing in the manual. One paragraph is devoted to linking. We think that this is hardly adequate. In the end we stole the link batch file LNK.BAT from the C-Systems work disk.

C86 supports the small and big models. The 8087 is also supported by a compiler option and a special 8087 library. Programs compiled with this option selected will not run on machines that do not have the 8087 coprocessor. Computer Innovations previously supported overlays, but with the addition of support for the big memory model, overlays were dropped.

Sieve compiled and ran fine but Fib wouldn't parse. It was then that we found out that the C86 manual lists no error messages. It only says that error messages "should be self-explanatory." We wouldn't have looked in the manual if that were true, honest. The problem with Fib was the *void* typing again. Removal of the offending *void* allowed Fib to compile and run.

Deref proved to be too complex again. Reducing the indirection to 14 levels proved to work. At that level of complexity, Deref compiled in 106 sec, ran in 6 sec, and took 12,680 bytes. Matrix required a *#include <stdio.h>* before it would compile.

Wizard C

There is nothing like a good name to get a product off the ground quickly, and we like the name Wizard. It makes us feel like real hackers. Wizard Systems' Wizard C seems to have been written by someone who has been in the business a while. This especially shows in the documentation, which is conversational without being preachy or snobbish.

Wizard C barely fits on a double-sided floppy. We were finally able to squeeze all of the required files onto the disk by deleting COMMAND.COM. This makes the disk unbootable but much easier to use during compilation.

Using a confusing set of compiler options, you can generate programs under the small, medium, or large model. The default is the small memory model. Adding the -e option gives you extended data, but this option must be used with the

Bundled software and essential information

	Manufacturer	Name	Price	Version	Operating system	Editor	Assembler	Linker	Librarian	Debugger	No. compiler phases	Optimizer pass	Compiler output	Full K&R	Full K&R storage classes	Compiler source included	Pages in manual
MS-DOS/PC-DOS compilers	Lattice	C	$500	2.13(2.14 avail)	MS-DOS, CP/M 86	no	N.A.	no	yes	no	2	no	OBJ	yes	yes	no	150+
	Microsoft	C	$500	2.03	MS-DOS	no	N.A.	yes	yes	no	2	no	OBJ	yes	yes	no	150+
	Microsoft	C	$450-500	3.00.13	MS-DOS	no	N.A.	yes	yes	no	4	yes	OBJ	yes	yes	no	832
	Mark Williams	MWC86	$500	2.0	MS-DOS, Coherent, CP/M 86, ISIS, RMX, VAX/VMS	no	yes	yes	yes	yes	4	yes	OBJ	yes	yes	no	238
	Rational Systems	Instant C	$495	.9(1.0 avail)	MS-DOS	yes	no	N.A.	N.A.	yes	0	N.A.	N.A.	no	yes	no	175
	Wizard Systems	Wizard C	$450	1.3(2.0 avail)	MS-DOS, TOS, VAX/VMS	no	no	no	no	no	3	no	OBJ or ASM	yes	yes	no	243
	Computer Innovations	Optimizing C86	$395	2.20G (2.20J avail)	MS-DOS, CP/M 86	no	no	no	yes	no	4	yes	OBJ or ASM	yes	yes	no	275+
	Supersoft	C	$350	1.2	MS-DOS, CP/M, CP/M 86	no	no	no	no	no	3	yes	ASM	no	no	no	228
	Manx Software	Aztec C/PRO	$498	2.20B	MS-DOS, CP/M 86, Macintosh	yes	yes	yes	yes	yes	1	no	ASM	no	yes	no	400+
	C-Systems	C	$199	2.06(2.08 avail 2/1)	MS-DOS	no	yes	no	no	yes	3	no	ASM	no	yes	no	193
	C Ware	DeSmet C	$159	2.4	MS-DOS, CP/M 86	yes	yes	yes	yes	yes	3	no	.0	no	no	no	128
	Datalight	Small-C	$40	2.1	MS-DOS	no	no	no	no	$40	1	no	ASM	no	no	yes	45
CP/M compilers	Ecosoft	Eco-C	$250	3.1(3.2 avail 1/1)	CP/M	no	yes	yes		no	4		ASM	no	yes	no	210
	Alcor	Alcor C	$139	2.01.00	CP/M, MS-DOS	yes	no	yes		no	1		p-code	yes	yes	no	486
	BDS	BDS C	$150	1.50a	CP/M, MP/M	no	N.A.	yes(2)		yes	2		.CRL	no	yes	no	184
	Code Works	Q/C	$95.00	3.2B(3.2C now avail)	CP/M, MP/M	no	no	no		no	1		ASM	no	yes	yes	173
	Software Toolworks	C/80	$49.95	3.1	CP/M, MP/M, MS-DOS	no	yes	no		no	1		ASM	no	yes	no	50
	Hendrix	Small-C	$25.00	2.1	CP/M, MP/M	no	no	no		no	1		ASM	no	no	yes	256
OS-9 compilers	Microware	C	$400	1.1.4	OS-9	no	yes	yes	no	ASM level	2	yes	ASM	no	yes	no	132
	Introl	C	$425	1.55	OS-9, Flex, Uniflex	no	yes	yes	yes	no	4	yes	ASM	no	no	yes	218
CP/M 86	Digital Research	C	$500	1.11	CP/M 86 MS-DOS	no	yes	yes	yes	no	2	no	OBJ	yes	yes	no	500+

N.A. = Not Applicable.

Table 1.

Software Development Tools

If:

Manufacturer-Compatible, Fully-Supported Cost-Effective, Cross and Native Development tools are important to you!

Then:

Let us tell you about MICROTEC RESEARCH's extensive line of field-proven software development tools for serious software developers.

C
Pascal
Assemblers
Simulators

Manufacturer Compatible

MICROTEC RESEARCH has been providing flexible and economical solutions for software developers since 1974. We've grown with the industry as our cross software development tools have transformed many large and small computers into powerful cross development systems for microprocessor applications. Our software tools are manufacturer compatible. Therefore, software made using manufacturers development systems may be more productively used in the MICROTEC RESEARCH cross-development environment.

Fully-Supported

MICROTEC RESEARCH's products are continually maintained and updated based upon the experience of thousands of installations worldwide. Revisions and modifications are distributed to licensees in warranty or covered by a service contract. Upgrades, which are major enhancements to a product's capabilities, are available at a significant discount. Our update/upgrade policy assures users they'll always be current with the fast moving world of microprocessor development.

Effective Differences

Beginning with product concept, through development, quality assurance, and post-sales support — **Quality, Compatibility,** and **Service** are the differences which set MICROTEC RESEARCH apart from the others.

Start Saving Time & Money

If you are a serious software developer, shopping for software development tools, call or write today for more information. **800-551-5554** In CA call **(408) 733-2919**

Target Microprocessors

- 8086/80186
- 8096
- 8080/8085
- 8051
- 8048
- Z8002
- Z80
- Z8
- NSC 800
- 9900...
- 68000/08/10
- 6809
- 6800/01/02
- 6805
- 6301
- 6305
- 6500
- G65SCXX, G65SC1XX
- R65C00, R6500/1..
- others

Host Computers

- DEC VAX, PDP-11
- DG MV-Series
- DG Eclipse
- Apollo
- UNIX Systems
- IBM PC
- Data General/ONE
- HP 150
- DEC Rainbow
- others

Software Tools

- C Compilers
- Pascal Compilers
- Assemblers
- Linking Loaders
- Librarians
- Conversion/ Download
- Communications
- VT-100 Emulator

for Serious Software Developers

MICROTEC RESEARCH

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
Mailing Address: P.O. Box 60337, Sunnyvale, CA 94088
(408) 733-2919 • Telex (ITT) 4990808

MICROTEC is a trademark of Microtec Research, Inc., Santa Clara, CA

CIRCLE 71 ON READER SERVICE CARD

-m option to create a large model program. The -m option used alone generates a medium model program. A nice feature of the compiler is that it automatically invokes the linker, reducing the need to babysit the compilation process.

One very useful program in the UNIX C programming environment is LINT. This is not something you pick off of your clothes but an extended error checking program designed to aid in writing code. Wizard C has LINT-type error checking built into the compiler. This feature produced several interesting warnings during compilation of the benchmark programs. In addition, Wizard C is supplied with the LINT program, and phase one of the compiler can be told to generate lint files (.LNT) to be used by LINT.

The Sieve and Fib benchmarks compiled and ran fine, but Deref struck once again. Wizard C came up with the very descriptive message "Disaster, Type stack overflow." The compiler could only handle 16 levels of indirection before the stack blew. At that level of indirection, Deref compiled in 72 sec, ran in 8 sec, and required 8,444 bytes of code. Matrix required a *#include <stdio.h>* before it would compile.

Datalight's Small-C

As an indication of MS-DOS's lineage, we find the migrant Small-C fresh from its appearance in the CP/M world. This is J. E. Hendrix's version 2.1 of Small-C, retargeted for the 8086 processor family by Datalight. Most of the words written later on about the CP/M version of Small-C apply to the MS-DOS version as well. Small-C for MS-DOS supports only the small memory model.

Small-C has a loyal group of followers, which is most evident from the number of programs written in the language appearing in *Dr. Dobb's Journal* over the past few years. Therefore, we find this version for PC-DOS and MS-DOS to be most welcome.

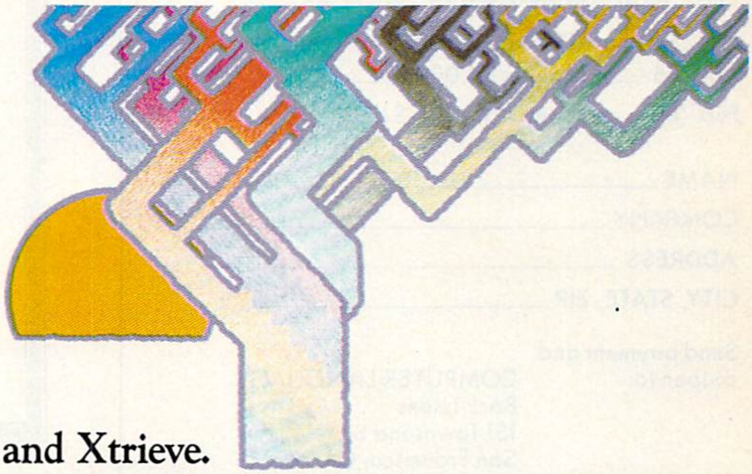
Small-C requires but is not supplied with an assembler and a linker. We used the Microsoft macro-assembler MASM version 1.0 and the linker supplied with DOS 3.0 in our tests. Two programs supplied with Small-C were batch file CC.BAT, which can be used to automate the compilation process, and LL.BAT, which does the same for linking. CC.BAT worked fine but LL.BAT would not work with version 2.2 of the DOS linker as supplied. This was due to the specification of the bit bucket NUL: as the target for the linker's list file. Apparently LINK version 2.2 (supplied with DOS 3.0) prefers NUL (without the :). A quick swipe of the editor and Small-C was off and running.

In the benchmark tests, Small-C compiled and ran Sieve with no problem but hit the rocks on the other three programs. It couldn't compile Fib until references to *void* and *unsigned* were removed. Then it was able to compile and properly run the benchmark. Deref requires structures and Matrix requires floating point arithmetic and two-dimensional arrays. Small-C does not support these three features so the last two benchmarks weren't compiled or run.

At $40, Small-C really isn't intended to be a heavy-duty development tool although it can be remarkably useful. It serves as a low-cost introduction to the C programming language and an instructive example of how to write a compiler. Small-C is the only MS-DOS C compiler we reviewed here which includes the source. The library source is also included. Thus you can learn how compilers are constructed and modify the existing Small-C compiler experimentally.

C Ware's DeSmet C

One rung up the price ladder from Small-C is a terrific package from C Ware called DeSmet C. For $159 you get a load of



Btrieve and Xtrieve.
B-tree file access for all your programming languages. With the query tool your users demand.

Introducing a powerful data base combination for PC application developers. Btrieve$^{\text{TM}}$, the most sophisticated file access method for your IBM$^{\text{TM}}$ PC. In single user and network versions. And Xtrieve$^{\text{TM}}$, a new menu-driven query tool that gives you—and your users—fast access to information.

Btrieve: for professional programmers. Btrieve provides fast, flexible file management for all your application development. All your programming languages—BASIC, Pascal, Cobol, C. With multikey access to records. Automatic file recovery. Unlimited records per file. Duplicate, modifiable, and segmented keys.

With Btrieve, you can develop *better* applications *faster.*

Xtrieve: easy window interface. The ideal complement to Btrieve, Xtrieve is the non-programmer's interface. Xtrieve's full relational capabilities let users define a virtual table of data from multiple files. Then Xtrieve speeds them through query building with a series of easy-to-follow windows.

No command language. Xtrieve is completely menu-driven, so there's no need to memorize command language. Or special syntax. Everything you need is on the screen.

Xtrieve features a full range of restriction criteria. Online help messages. And interfaces to access information from Lotus 1-2-3$^{\text{TM}}$ and dBase II® files.

Network capabilities. Network versions of Btrieve and Xtrieve allow data sharing in PCnet$^{\text{TM}}$, NetWare$^{\text{TM}}$, EtherSeries$^{\text{TM}}$, MultiLink$^{\text{TM}}$, and OmniNet$^{\text{TM}}$ LANs.

For more information or to order, contact:

SoftCraft Inc.
P. O. Box 9802 #590 Austin, Texas 78766 (512) 346-8380

Suggested retail prices: Btrieve, $245; Xtrieve, $195. Btrieve/N (network), $595; Xtrieve/N, $395. Dealer inquiries welcome. Btrieve requires PC-DOS or MS$^{\text{TM}}$-DOS 1.X or 2.X; Xtrieve, PC-DOS or MS-DOS 2.X.

Btrieve and Xtrieve, IBM, 1-2-3, dBase II, PCnet, NetWare, EtherSeries, MultiLink, OmniNet, and MS are trademarks of SoftCraft Inc., International Business Machines, Lotus Development Corp., Ashton-Tate, Orchid Technology, Novell Data Systems, 3Com Corp., Davong Systems Inc., Corvus Systems, and MicroSoft Inc.

CIRCLE 78 ON READER SERVICE CARD

ORDER BACK ISSUES OF COMPUTER LANGUAGE!

Premier ____ copies × \$4.00 = \$_______
Oct. '84 ____ copies × \$4.00 = \$_______
Nov. '84 ____ copies × \$4.00 = \$_______
Dec. '84 ____ copies × \$4.00 = \$_______
Jan. '85 ____ copies × \$4.00 = \$_______
Feb. '85 ____ copies × \$4.00 = \$_______

NAME ____________________
COMPANY ____________________
ADDRESS ____________________
CITY, STATE, ZIP ____________________

Send payment and coupon to:

COMPUTER LANGUAGE
Back Issues
131 Townsend St.
San Francisco, CA 94107

WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. Dealer inquiries invited. WRITE is \$239.00.

BDS's C Compiler

This is the compiler you need for learning the C language and for writing utilities and programs of all sizes and complexities. We offer version 1.5a, which comes with a symbolic debugger and example programs. Our price is (postpaid) \$130.00.

Tandon Spare Parts Kits

One door latch included, only \$32.50.
With two door latches \$37.50.
Door latches sold separately for \$7.00.

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Workman & Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

WORKMAN & ASSOCIATES USER-FRIENDLY SOFTWARE

CIRCLE 68 ON READER SERVICE CARD

UNIX TIMESHARE+

*UNIX System III POWER and sophistication are yours. Let THE SOLUTION turn your micro into all you dreamed it could be, bringing the Ultimate programming environment as close as your modem. Just a local call from over 300 cities nationwide via Telenet

THE SOLUTION™

- **EXPANSIVE SOFTWARE DEVELOPMENT FACILITIES** including Language and Operating System design.
- **LANGUAGES:** C, Fortran 77, RATFOR, COBOL, SNOBOL, BS, Assembler + Artificial Intelligence programming via LISP.
- **USENET Bulletin Board System**—800 + international UNIX sites feeding over 190 categories, typically bringing you more than 160 new articles per day.
- **Interuser and Intersystem mail** + 'chat' capability.
- **UNIFY:** Sophisticated data-base management system.
- **UNIX & System enhancements** from U.C. Berkeley and Korsmeyer Electronic Design Inc.
- **Online UNIX manuals** + Expert consultation available.
- **SOLUTION-MART:** Hardware/Software discount shopping database.
- **LOW COST and FAST** response time. (as low as \$8.95 hr. connect time + \$.05 cpu sec. non-prime)
- **\$24.95 = 1 hr. FREE system time** + SOLUTION News subscription + BYTE BOOK (Introducing The UNIX System 556 pp.).

*UNIX is a trademark of Bell Labs.

VISA MasterCard
Payment via VISA or Master Card

Korsmeyer
ELECTRONIC DESIGN, INC.

5701 Prescott Avenue
Lincoln, NE 68506-5155
402/483-2238
10a-7p Central

CIRCLE 34 ON READER SERVICE CARD

Benchmark results

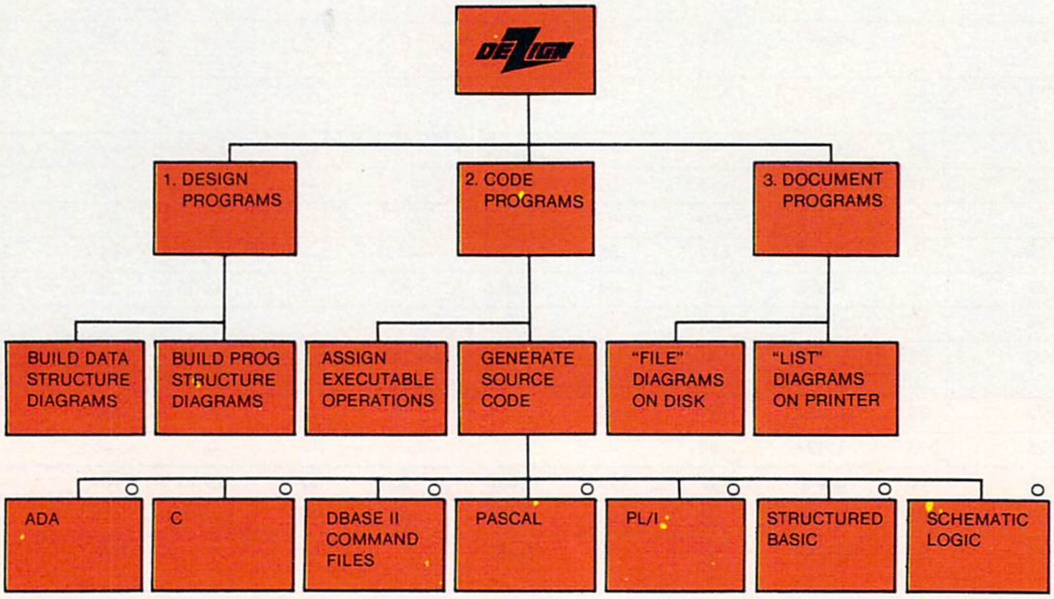
	Compiler	Sieve			Fib			Deref			Matrix		
		Compile time (sec)	Run time (sec)	Size (bytes) (sec)	Compile time (sec)	Run time (sec)	Size (bytes) (sec)	Compile time (sec)	Run time (sec)	Size (bytes)	Compile time	Run time	Size (bytes)
MS-DOS/PC-DOS compilers	Lattice	98	11	21,902	80	58	13,710	90	13	13,726	163	29	25,176
	Microsoft (2.03)	103	11	19,754	81	57	11,578	90	13	11,578	99	28	22,722
	Microsoft (3.0)	102	11	6,926	109	72	6,924	104	10	6,940	—	—	—
	Mark Williams	79	12	6,887	81	—	6,869	79	11	6,874	107	29	10,847
	Rational Systems	4	41	640	3	176	640	—	—	—	8	35	1,408
	Wizard Systems	74	12	16,490	76	52	8,444	—	—	—	84	16	18,826
	Computer Innovations	111	13	12,729	112	46	12,728	—	—	—	134	27	13,766
	Supersoft	132	28	13,112	114	47	4,738	119	15	4,822	—	—	—
	Manx 1.06D	63	17	4,448	58	82	4,384	—	—	—	94	21	7,920
	Manx 2.20B	64	11	4,448	61	61	4,448	—	—	—	92	16	7,840
	C-Systems	119	19	27,716	112	58	19,456	—	—	—	191	22	31,114
	C Ware	49	12	6,656	48	44	6,656	49	11	6,656	55	17	7,168
	Datalight	94	40	14,182	82	76	5,861	—	—	—	—	—	—
	Digital Research	108	12	27,136	109	43	18,944	145	14	18,944	145	18	31,872
CP/M compilers	Ecosoft	93	39	14,464	39	83	—	78	15	—	154	18	—
	Alcor	25	215	1,024	30	—	—	—	—	—	—	—	—
	BDS	25	48	3,840	27	113	3,840	25	12	3,712	—	—	—
	Code Works	67	48	12,160	67	104	4,352	46	12	4,352	—	—	—
	Software Toolworks	117	26	12,288	114	57	3,584	—	—	—	172	20	12,288
	Hendrix	270	58	16,640	—	—	—	—	—	—	—	—	—
OS-9 compilers	Microware (from disk)	111	15	3,873	111	54	3,812	—	—	—	151	43	8,207
	Microware (from RAM)	72	12	3,873	74	12	3,812	—	—	—	112	40	8,207
	Introl (from disk)	195	17	7,618	195	34	7,546	191	14	7,567	246	29	8,370
	Introl (from RAM)	—	11	7,618	—	28	7,546	—	8	7,567	—	22	8,370
CP/M-86 compilers	Digital Research (CP/M-86)	158	12	28,672	155	43	20,480	153	14	20,480	195	18	34,816
	Digital Research (MS-DOS)	108	12	27,136	109	43	18,944	145	14	18,944	145	18	31,872

Table 2.

EVERY PROGRAMMER/ANALYST NEEDS . . .

DEZIGN

. . . THE STRUCTURED PROGRAMMING TOOL!


DEZIGN
1. DESIGN PROGRAMS
2. CODE PROGRAMS
3. DOCUMENT PROGRAMS
BUILD DATA STRUCTURE DIAGRAMS
BUILD PROG STRUCTURE DIAGRAMS
ASSIGN EXECUTABLE OPERATIONS
GENERATE SOURCE CODE
"FILE" DIAGRAMS ON DISK
"LIST" DIAGRAMS ON PRINTER
ADA
C
DBASE II COMMAND FILES
PASCAL
PL/I
STRUCTURED BASIC
SCHEMATIC LOGIC

Design your programs right on the screen using modern techniques based on the popular Jackson Structured Programming method (JSP)!

DEZIGN is more than just another flowcharting tool. It is an integrated tool for designing and documenting programs and for generating source code in any of the languages listed above!

DEZIGN enables a software developer to create Data and Program Structure Diagrams using the Sequence, Selection, and Iteration constructs; assign detailed statements to the Diagrams; and synthesize source code from the control logic represented on the Diagrams and the detailed statements assigned to them.

DEZIGN lists for $200 and is available directly from Zeducomp, from IBM and Zenith personal computer dealers, and from software retailers.

- DEZIGN-PC runs under DOS 2.0 and 2.1.
- DEZIGN-86 runs under CP/M-86 1.1 and Concurrent CP/M-86.
- DEZIGN-16 runs under ZDOS on Zenith Z-100 computers.

"Principles of Program Design," by M. Jackson, Academic Press, describes the JSP metholology and is available for $50 directly from Zeducomp and from most professional/technical book stores.

ZEDUCOMP

P.O. BOX 68 ● STIRLING, NJ 07980 ● (201) 755-2262

dBASE II is a trademark of Ashton-Tate, Inc.

CIRCLE 91 ON READER SERVICE CARD

software—a complete development system, in fact. You start with a screen editor geared to the IBM PC. You can pay well over $159 for just an editor and we have just begun.

Next is the compiler. It isn't a small compiler at all, but a nearly full K&R implementation. What's missing? Well, some of the data types, preprocessor commands, storage classes and an operator or two. But these omissions didn't stop DeSmet C from compiling and running the four benchmarks with no modifications!

The compiler supports only the small memory model but makes up for the memory limitations by supporting overlays. Support is also available for the Intel 8087 numeric coprocessor. Two libraries are supplied with DeSmet C. One supports floating point calculations with assembly language routines and the other uses the 8087.

Also included in this software bargain bonanza are a binder (linker), an assembler, a performance profiler (!) and a librarian. The linker is necessary because the DeSmet compiler output is not the .OBJ file the DOS linker can use. The assembler is not needed for the compilation of C programs. It is a parallel development tool for routines that must be coded in assembly language for efficiency. The assembler produces object files compatible with the DeSmet binder.

Do you know what a profiler is? It is a jolly little utility that uses the IBM PC's real-time clock to generate a histogram of your program's operation. It does this by latching on to a periodic interrupt and sampling the instruction pointer (IP) at each interrupt. The value in the IP is used to increment one of 1,024 counters or "buckets". The result is a good approximation of where your program spends its time. Current versions of DeSmet C allow the profiler to work symbolically.

We did not have a chance to work with the DeSmet debugger or librarian, but they would appear to be most adequate from the documentation. Speaking of the documentation, we get to our only complaint about DeSmet C. For $159, we think C Ware could afford a vinyl 3-ring binder. The documentation is good, but it needs packaging.

Finally, DeSmet C was the easiest DOS C compiler to bring up. Twenty minutes out of the box, it compiled a program. That includes the time it took to create working copies of the compiler and binder. It worked just like the instructions said it would too! Truly a miraculous package.

C-Systems C

Returning to earth, we look at C-Systems C. This compiler is more a complete implementation of K&R although still not a full version. C-Systems C does not come with an assembler or linker. Using the DOS assembler and linker, the complete development package of compiler, assembler, linker and library won't fit on one 360K floppy disk. Thus you will need a compile disk and a link disk and will be swapping between them unless you have a hard disk or RAM disk.

The compiler supplied for $199.00 supports the small and big memory models. Libraries for the compact and medium models are $50.00 each. Library source is not included but is available at an extra charge. The 8087 is also supported. It can be sensed automatically or the compiler can be instructed to assume that an 8087 will be present, for maximum performance.

This compiler did not work the way the manual said it would. A batch file called CC.BAT is supplied to run the compilation. You are supposed to have your source on drive B and the compile disk on drive A. Then, while logged on to B, type "a:cc <filename>" and your file will compile. Except it won't. CC.BAT expects the compiler files to be on the default drive, which in this case is B, but the files are actually on A. Logging onto drive A and typing "CC B:<filename>" works ever so much better. You'd think they would have tried it, wouldn't you? A batch file for linking, called LNK.BAT, did work as advertised.

Another problem we had in running C-Systems C the first time was due to our own error, with plenty of help from C-Systems. They supply batch file CC.BAT and an executable file called CC.EXE on the same disk, and we copied them both over to our working disk without looking. CC.EXE doesn't perform the same as CC.BAT, yet it overrides it when CC is called from DOS. We shouldn't have copied CC.EXE to our working disk and neither should you unless you know what it does. Our opinion: C-Systems should not supply its programs with this potential for error built in.

Though apparently a more complete implementation of C than DeSmet, C-Systems C had problems in the benchmarks. Sieve compiled and ran fine, but problems cropped up getting the other three to compile. C-Systems C didn't like *void* in Fib, couldn't handle 40 levels of indirection in Deref, and required a *#include <stdio.h>* in Matrix.

Since the compiler couldn't deal with the indirection in Deref, we started reducing the level of complexity until the program would compile. The magic number was 30 levels of indirection. No values were entered in the benchmark table for this program since it is not the full program, but at 30 levels of indirection, Deref compiled in 106 sec, ran in 9 sec, and linked to 19,426 bytes.

VEDIT PLUS

Easy to Use
Word Processor with the Power of BASIC

VEDIT has been acclaimed for the last four years the industry standard in text editing.

InfoWorld
Software Report Card
VEDIT 1.36

	Poor	Fair	Good	Excellent
Performance	☐	☐	☐	☒
Documentation	☐	☐	☐	☒
Ease of Use	☐	☐	☒	☐
Error Handling	☐	☐	☐	☒

Now there's VEDIT PLUS.

VEDIT PLUS is easy to learn, yet can do things far beyond ordinary word processors:

- Sort a mailing list
- Perform arithmetic computations
- Compare files
- Edit mulitple files of unlimited size

The power of VEDIT PLUS is equal to a high-level language - it gives you:

- If-then-else decision making
- Pattern matching
- User prompts and input
- An optional Z80 to 8086 translator

Expect a lot from VEDIT PLUS. It's from CompuView - nationally recognized for user support.

For a demo disk, more information or the dealer nearest you call toll free:

1-800-327-5895

VEDIT PLUS is available for practically every microcomputer . . .$225

CompuView
PRODUCTS, INC.

1955 Pauline, Ann Arbor, Mi 48103
(313) 996-1299 • (800) 327-5895

CIRCLE 40 ON READER SERVICE CARD

Benchmarks

The suite of four benchmark programs we used in reviewing the C languages was taken from a recent article in *BYTE* magazine.[1] Written by Dr. David D. Clark, these benchmarks test out various features of the languages.

First, however, we found it necessary to debug the benchmarks. Something must have happened to Clark's programs between benchmark testing and publication because two of the programs listed wouldn't run. If anybody would like a copy of our reworked benchmarks, written in C, they are all available on the *COMPUTER LANGUAGE* Bulletin Board Service and on this magazine's account on CompuServe. Look for the files BENMRK.*. With all the tables and lengthy text from this review, there was just not enough room in the magazine left to print these benchmarks.

The first benchmark we used is the ever popular Sieve of Eratosthenes, written in a very simple dialect of C. This program is also called the prime number generator. It is a good test of a language's ability to run loops since the program has a triply nested loop. Sieve computes all prime numbers from 3 to 16,381. It does this by manipulating an array of 8,191 flags, with each flag representing an odd number from 1 to 16,381. Even numbers are obviously not prime and thus require no flags in the array. At the completion of the program, the largest prime found (16,381) is printed as is the number of primes (1,899). All compilers were able to run this program and obtain the correct answer.

The second benchmark, called Fib, generates Fibonacci numbers. This is a recursive algorithm so the program tests the speed of function calls and the ability of the language implementation to perform recursion. The Fibonacci numbers are computed using the following algorithm:

```
If $x <= 2$ then $F(x) = 1$
If $x > 2$ then $F(x) = F(x-1) +$
    $F(x-2)$
```

When the program completes, it prints the value of *F(24)*, which is 46,368.

Though our third benchmark is a very short program, many of the compilers had a lot of trouble with it. Called Deref, it tests the speed at which a C deals with indirection. By setting up a nested structure, the program repeatedly accesses a variable through 40 levels of indirection. This complexity of indirection was managed by only a few compilers.

The fourth benchmark was derived from a program proposed by *BYTE* columnist Jerry Pournelle.[2] Called Matrix, it tests floating point arithmetic speed. Since some of the compilers reviewed in this article do not implement floating point arithmetic, they weren't able to compile this benchmark. C programs don't always require floating point calculations so the inability to run this benchmark is not a terrible blot on a compiler's record.

Still, it is valuable to compare floating point performances on those C implementations that can run this program. The program creates three matrices. It then fills (initializes) the matrices and computes values by iteratively performing matrix multiplication and addition. This program is designed to be very compute-intensive. The final value printed is the sum of all elements in the C matrix, which is 465,880.

Rational Systems' Instant C

And now for something completely different, as Monty Python would say. Instant C from Rational Systems is a C interpreter! We felt right at home with Instant C. It has the look and feel of C and the texture of a BASIC interpreter, all at the same time.

Imagine instant gratification from this language. Type "i++" and the value *i* is printed, then incremented. Type "++i" and *i* is incremented, then displayed. Learn C the easy way, with Instant C. We debugged all of the DOS benchmark programs in Instant C before unleashing the nasty old compilers on them.

The reason we did this is easy to understand by looking at the compile times in the benchmark results table—not a compile (load) time over 8 sec. But not all can be inferred from this table. The incredible (200K!) interpreter includes an editor. If you load a program with syntax errors, the interpreter will instantly switch to the edit mode and plunk the cursor on the offending character. You won't be able to exit this mode until you clean up all of your syntax errors or cry uncle (quit).

Editing is very slick. Type "ED <function name>" and only the function appears, while the interpreter switches to edit mode—modular text editing in the context of an interpreter. Some very admirable ideas are written into this little piece of code.

How about those error messages? "I'm sorry, but I don't know the word 'number'." appeared when we had used "number" in one place and "NUMBER" in another. Being a compiler means never having to say you're sorry, but the interpreter seems to have that ability.

The bad news shows in the execution time column in the benchmark results table. Interpreters are slower than compilers by a factor of two or three. Yet look at the Matrix execution time. The interpreter approaches the speed of the compilers in floating point intensive programs. This is somewhat intuitive, as a floating point routine is the same the whole world over, compiler or interpreter.

Deref was again the heartbreaker. Above 16 indirection levels, the syntax is too complex for Instant C's parser. At 16 indirection levels, the program ran, but the memory-resident source was destroyed. At 14 levels, all was well.

As to the source being clobbered—the copy of Instant C we received was a prerelease copy, version 0.9. This problem will no doubt be fixed for the released version. Let's hope so, this program is too good to do without!

Unfortunately, you may have to do without it. At $500, you get a great tool for writing and testing functions and modules and nothing to compile them after they are working. Rational Systems recommends that you purchase someone else's compiler as a companion to Instant

C, but we think the combined purchase price is a bit steep.

On the other hand, if Instant C could make you twice as productive, just think what your company will save not having to hire a second programmer. We sincerely feel that Instant C can have a major positive impact on programmer productivity.

Mark William's MWC86

From the land of the minicomputer and the super-microcomputer comes Mark Williams Co's MWC86. Originally from the UNIX environment (Coherent from Mark Williams), MWC86 is a complete development system for the MS-DOS environment. It includes a compiler, linker, assembler, librarian and symbolic debugger.

MWC86 supports both small and large models. In addition, floating point libraries for non-8087 and 8087 use are included. The compiler, linker, and one library fit on a single disk.

The manual looks nice. It is typeset and has fancy tab dividers. However, it is not well indexed and doesn't seem to have any error message explanations. The index includes only function names and no topics, so it is almost useless for the beginner. The lack of error message explanations is completely unforgivable.

The csd C-source debugger is this package's claim to fame. Included is a vacuum-formed template that fits over the IBM PC's special function keys. The csd debugger completely disassociates the debug process from the assembly language level, allowing the programmer to deal only with C source code. The debugger allows the program to run at full speed, and program size does not grow because the debugger is being used. On-line help screens are also available, making this the most comprehensive debugger we have seen for MS-DOS C compilers.

During compilation of the benchmark programs, we encountered several errors and warnings. As mentioned previously, these were not explained in the manual and we were left to our own resources. Ultimately, we were able to compile all four of the benchmarks, but Fib did not run. In fact it hung the machine, making us suspect MWC86's capacity for recursion.

Microsoft C (Lattice version)

It is somewhat common knowledge that Microsoft C is actually Lattice C (reviewed later). This isn't a secret as the Microsoft manual lists Lattice as joint copyright holder. Since the two compilers sell for the same price ($500), it is very interesting to compare and contrast the two packages.

Microsoft C was supplied as version 2.03. This compiler supports the small, program, data and large models. The Microsoft C compiler does not support the 8087, but data formats have been made to conform to the 8087 definitions.

No help is supplied for the creation of the working disk. This is apparently left as an exercise for the user. Microsoft supplied all files on single-sided disks so there was quite a bit of disk swapping needed to create the work disks. No batch files were supplied to aid in compilation or linking, but sufficient information was in the manual for an experienced programmer to create these files. A copy of K&R was included in the documentation.

The Microsoft C compiler is a full C implementation. It was able to compile and run the benchmarks with two modifications. The *void* typing in Fib had to be eliminated, and Matrix required the addition of *#include <stdio.h>*. In addition, a fluke collision occurred between the array *c[][]* and a procedure called *c* in the standard program entry module. This

PROGRAMMING TOOLS
for Pascal and C

*C Productivity Series—
The Professional's Edge*

Blaise Computing has a range of programming aids for the most popular C compilers in the IBM environment that no serious system developer should be without. These packages help you to easily access advanced capabilities of the hardware and operating system, and to finish your projects with a substantial saving of time and effort. With software development costs and pressures as great as they are, can you afford not to take advantage of the finest tools available?

- **C TOOLS™** puts advanced string handling functions at your disposal and provides a high-level interface to all BIOS functions from your C program. Complete screen handling, graphics primitives, and a substantial group of useful, general-purpose functions are also featured. **$125**
- **C TOOLS 2™** lets your program perform all the advanced DOS 2.0 services. Program chaining, software interrupt handling, and dynamic memory allocation are all done "right." Buffer and file handling functions are provided, as well as a general DOS gate. **$100**
- **C VIEW MANAGER™** is our display screen management system that makes screen development and documentation much faster. It comes with a complete library of C functions which use the screens you have developed to recall and display information, capture and validate field data entry, and provide context-relevant help files. **$275**
- **ASYNCH MANAGER™** is a library of interrupt-driven routines providing a general interface to both COM ports for your asynchronous communications applications. Introductory price of **$175** includes all source.

All of these products may be used by developers with no royalty payments to Blaise Computing. Source code either comes with the package, or is available. We support Lattice, Computer Innovations, and Microsoft C compilers. To expedite your order or to obtain further information, call or write us directly.

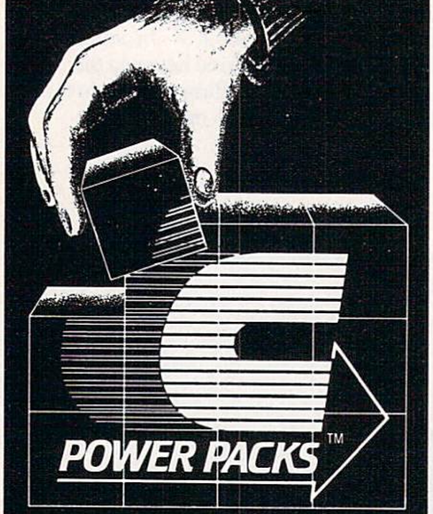
Blaise Computing's Programmer Productivity series is also available in versions for the Pascal language.

◆

BLAISE COMPUTING INC.
2034 Blake Street Berkeley, CA 94704
(415) 540-5441

CIRCLE 11 ON READER SERVICE CARD

"This is a beautifully documented, incredibly comprehensive set of C Function Libraries."
— Dr. Dobb's Journal



COMPLETE SOURCES

■ **PACK I: Building Blocks I** 250 Functions: DOS, Printer, Video, Asynch	$149
■ **PACK 2: Database** 100 Functions: B-Trees, Variable Records	$399
■ **PACK 3: Communications** 135 Functions: Smartmodem™, Xon/Xoff, Modem-7, X-Modem	$149
■ **PACK 4: Building Blocks II** 100 Functions: Dates, Text Windows, Pull-down Menus Data Compression	$149
■ **PACK 5: Mathematics I** 35 Functions: Log, Trig, Square Root	$99
■ **PACK 6: Utilities I** Archive, Diff, Replace, Scan, Wipe (Executable Files only)	$99

Lattice™, Microsoft™, DeSmet™, CI-86™ Compilers on IBM PC/XT/AT™
Small and Large Memory Models.
Credit cards accepted
($7.00 handling/Mass. add 5%)

SOFTWARE HORIZONS INC.

165 Bedford Street
Burlington, Mass. 01803
(617) 273-4711

NOVUM ORGANUM

CIRCLE 25 ON READER SERVICE CARD

resulted in a doubly-defined symbol error. Changing the array name to *qc* quickly solved this problem.

Lattice C

It is apparent that Lattice C version 2.13 is a later version of the compiler sold as Microsoft C, version 2.03. Whole sections of the manual are duplicated in both versions. Lattice C also supports the four memory models: small, program, data, and large. Several features have been added to the Lattice version since Microsoft obtained the compiler.

The first addition is 8087 support. Lattice C will automatically sense the presence of the 8087 coprocessor and use the appropriate routines. Another addition is in expansion of the library to include more extensive mathematical functions. A *fork/wait* capability has been added which allows a program to create a child process and wait for it to complete.

One addition we particularly appreciated was an executive program called LC, which manages the compilation of programs. This replaced the batch file we created for Microsoft C. Unfortunately, no corresponding program is supplied for linking, so we used the file developed for Microsoft C, with suitable modifications.

We found the Lattice C manual to be quite complex. It is haphazardly assembled in three sections. The first section is a supplement containing the additions in version 2.10 of the compiler. The large middle section is the functional description of Lattice C, revision 2. The last section is a set of technical bulletins updating the first section supplement with the additions made in versions 2.10, 2.11, and 2.13. Finding information in this confusing array of pages is not very easy. Unlike the Microsoft compiler, Lattice does not include a copy of K&R with its manual.

Ignoring the manual, Lattice C is the most comprehensive compiler we have reviewed for MS-DOS. Its memory models, 8087 and library support provide a very full-featured compiler. If Lattice only supplied a debugger, it would be a most complete development system.

Lattice C was able to compile all of the benchmark programs, but Fib required the deletion of the *void* typing, and Matrix required the addition of *#include <stdio.h>*.

While linking Matrix we encountered one problem—we got several error messages for missing functions. The manual was not much help at first. Eventually we discovered that Lattice had decided to split the library, giving floating point functions a separate library. Changing the link command cleared this problem up. The scattered documentation on the version 2.13 revisions were at the root of this problem, as the other two manual sections were quite misleading.

Microsoft C 3.0

This compiler is not at all like the C compilers previously offered by Microsoft. Before release 3.0, Microsoft sold repackaged versions of the Lattice C compiler. With the release 3.0 package, Microsoft introduces a homegrown compiler with very powerful features that take very good advantage of MS-DOS. Of course, this is to be expected since MS-DOS is also a Microsoft product.

Microsoft has tried to reproduce a UNIX-like environment with the release 3.0 C compiler. This can be seen in the compiler environment and the library functions supplied. Nothing has been retained from the Lattice package. Microsoft does acknowledge its previous compilers, however, by supplying a conversion aid, a file called V2TOV3.H.

The release 3.0 package is very large. It has a four-phase compiler, a special linker, several libraries, a librarian, and a large collection of include files. The manual is very large and comprehensive. It is composed of three sections: a user's guide, run-time library reference, and language reference.

The manual recommends a two-disk partition for floppy disk systems, placing the compiler and *include* files on one disk, with one set of libraries and the linker on the second. Unfortunately, we couldn't squeeze all of the *include* files on one disk with the compiler. They must have gotten somewhat larger since the manual was written. The end result is that we highly recommend a hard disk for use with this compiler.

Microsoft has abandoned the Lattice memory models in favor of the more popular Intel-style models. Release 3.0 supports small, medium, and large memory models. In addition, Microsoft C release 3.0 supports the keywords *near* and *far*, which can override the selected memory model for data items. Memory model selection is made by compiler command line options.

We received a prerelease version of this compiler from Microsoft. It was able to compile the first three benchmarks with no problems, but the Matrix benchmark produced an internal compiler error. This bug has been reported to Microsoft, and we hope it will be fixed by the time the compiler is offered for sale. The release 3.0 compiler generated fairly compact code that ran quickly for the Sieve and Deref tests but suprisingly slow in the Fib test. In all, we found the new Microsoft compiler to be quite impressive and expect it to be a contender in the crowded MS-DOS C compiler market.

DSD80

FULL SCREEN SYMBOLIC DEBUGGER

"THE SINGLE BEST DEBUGGER FOR CP/M-80. A TRULY AMAZING PRODUCT."

LEOR ZOLMAN
AUTHOR OF BDS C

- ☐ Complete Upward Compatibility with DDT
- ☐ Simultaneous instruction, register, stack & memory displays
- ☐ Software in-circuit-emulator provides write protected memory, execute only code and stack protection.
- ☐ Fifteen single keystroke commands
- ☐ Thirty day money back guarantee
- ☐ On-line help & 50 page user manual

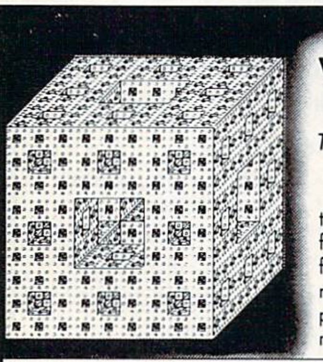
ONLY $195.
IBM PC VERSION AVAILABLE SOON!

SOFTADVANCES

P.O. BOX 49473 AUSTIN, TEXAS 78765 (512) 478-4763

MasterCard VISA

CIRCLE 77 ON READER SERVICE CARD



WALTZ LISP(TM)

The one and only ***adult*** *Lisp system for CP/M users.*

Waltz Lisp is a very powerful and complete implementation of the Lisp programming language. It includes features previously available only in large Lisp systems. In fact, Waltz is substantially compatible with *Franz* (the Lisp running under *Unix*), and is similar to *MacLisp*. Waltz is perfect for Artificial Intelligence programming. It is also most suitable for general applications.

Much faster than other microcomputer Lisps. • Long integers (up to 611 digits). Selectable radix • True dynamic character strings. Full string operations including fast matching/extraction. • Flexibly implemented random file access. • Binary files. • Standard CP/M devices. • Access to disk directories. • Functions of type lambda (expr), nlambda (fexpr), lexpr, macro. • Splicing and non-splicing character macros. • User control over all aspects of the interpreter. • Built-in prettyprinting and formatting facilities. • Complete set of error handling and debugging functions including user programmable processing of undefined function references. • Virtual function definitions. • Optional automatic loading of initialization file. • Powerful CP/M command line parsing. • Fast sorting/merging using user defined comparison predicates. • Full suite of mapping functions, iterators, etc. • Assembly language interface. • Over 250 functions in total. • The best documentation ever produced for a micro Lisp (300+ full size pages, hundreds of illustrative examples).

Waltz Lisp requires CP/M-86 or CP/M 2.2, Z80 and 48K RAM (more recommended). All common 5" and 8" disk formats available.

ProCode(TM)
INTERNATIONAL

15930 SW Colony Pl.
Portland, OR 97224

Unix* Bell Laboratories.
CP/M* Digital Research Corp.

Version 4.4
(Now includes Tiny Prolog written in Waltz Lisp.)

$169*

*Manual only: $30 (refundable with order). All foreign orders: add $5 for surface mail, $20 for airmail. COD add $3. Apple CP/M and hard sector formats add $15.

Call free **1-800-LIP-4000** Dept. #13
In Oregon and outside USA call 1-503-684-3000

CIRCLE 60 ON READER SERVICE CARD

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support.
  Full range transcendentals
  (tan, sin, cos, arctan, logs and exponentials)
  Data type conversion and I/O formatting.
- High level interrupt support.
  Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS/FORTH

- Fully Optimized & Tested for:

IBM-PC	IBM-XT	IBM-JR
COMPAQ	EAGLE-PC-2	
TANDY 2000	CORONA	
LEADING EDGE		

  (Identical version runs on almost all MSDOS compatibles!)
- Graphics & Text
  (including windowed scrolling)
- Music - foreground and background
  includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler
  (interactive, easy to use & learn)
- Compare

BYTE Sieve Benchmark		jan 83
HS/FORTH 47 sec	BASIC	2000 sec
w/AUTO-OPT 9 sec	Assembler	5 sec
other Forths (mostly 64k)		70-140 sec

FASTEST FORTH SYSTEM AVAILABLE.
TWICE AS FAST AS OTHER FULL MEGABYTE FORTHS!
(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: $250.

Visa Mastercard
Add $10. shipping and handling

HARVARD SOFTWORKS

PO BOX 2579
SPRINGFIELD, OH 45501
(513) 390-2087

CIRCLE 47 ON READER SERVICE CARD

FINALLY...

THE C JOURNAL

The C Journal will help YOU use C on YOUR machine —IBM PC™, CP/M™, Macintosh™, or UNIX™-based — micro, mini, or mainframe.

It's the ONE publication for programmers, software managers, and other computer professionals who need to keep aware of developments in the industry's fastest-growing language.

- regular columns for novice through advanced C programmers
- software product and book reviews
- tips on working with major compilers and operating systems
- news from the ANSI standards committee and the industry

For **FREE** sample issue and discount subscription information, write, call, or circle our reader service number. **The C Journal** is a quarterly publication, and costs $28/year (add $9 for overseas airmail).

Subscriptions/Advertising:	Editorial:
Christina Gardner	Rex Jaesche
(201) 989-0570	(703) 860-0091

another independent publication from

InfoPro Systems
3108 Route 10
P.O. Box 849
Denville, NJ 07834

MasterCard
VISA

CIRCLE 5 ON READER SERVICE CARD

TRSDOS TOOLS
MODELS II, 12, 16

TRS/C C COMPILER

Full K&R with source to the function library. UNIX compatible.	$85.00

ZSPF EDITOR

SPF, the choice of most mainframe programmers, is now available for Z80 machines. And it's panel driven so you can customize it!	$75.00

MODEL 100 C COMPILER

Now you can write efficient programs for your TRS-80 model 100 with ease. Or, learn the essentials of C programming while traveling!

C/100 - THE "PORTABLE" C COMPILER

Cassette version	$49.00
Disk/Video interface version	$59.00
Model II version (run on mod II,then download object code to model 100)	$79.00
Model III version (as above, for Mod III)	$79.00

Write or call for information on other TRS-80 software.

TRSDOS and TRS-80 are trademarks of Tandy Corp.

business utility software
109 minna ste 423 san francisco ca 94105

(415) 397-2000

CIRCLE 17 ON READER SERVICE CARD

OVERCOME
FORTRAN and PASCAL
LIMITATIONS WITH

NO LIMIT

$89 Visa/MC

A library of 58 Assembler routines transforms MS FORTRAN and PASCAL plus other 8086/87/88 FORTRANs into the flexible, responsive, complete language needed for the microcomputer environment. Featuring:

EXTENSIVE GRAPHICS	(Get, Put, Paint, Color, Dot, Line, Box, Circle, Ellipse, Large Characters)
FULL SCREEN CONTROL	(Windows, Cursor, Read/Write Screen)
STRING MANIPULATION	(Match, Compare, Concatenate/Extract, Pack, Justify, Zero Fill)
KEYBOARD CONTROL	(Read Key During Execution, String Read With Edit)
FILE MANAGEMENT	(Exist?, Rename, Delete)
COMMUNICATIONS	(Set Com Line, Send/Receive, Line/Modem Status)
OTHER FEATURES	(Peek, Poke, Determine Time/Date, Random Numbers, Beep, Clear Screen, OR/AND/XOR/NOT/NEG of Byte/Word, Printer Status)

For routines to meet language specific problems such as **Chaining** or **Command Line Read** call:

M | E | F Environmental Inc. P.O. Box 26537
Austin, Texas 78755 (512) 251-5543

CIRCLE 55 ON READER SERVICE CARD

OPT-TECH SORT™

SORT/MERGE program for IBM-PC & XT

Now also sorts dBASE II files!

- Written in assembly language for **high performance**
  Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. **COMPARE!**
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Handles variable and fixed length records
- Supports all common data types
- Filesize limited only by your disk space
- Dynamically allocates memory and work files
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use, includes on-line help feature
- Full documentation — sized like your PC manuals
- **$99** —VISA, M/C, Check, Money Order, COD, or PO
  Quantity discounts and OEM licensing available

To order or to receive additional information write or call:

OPT-TECH DATA PROCESSING
P.O. Box 2167 Humble, Texas 77347
(713) 454-7428

Requires DOS, 64K and One Disk Drive

CIRCLE 66 ON READER SERVICE CARD

COMPILERS FOR CP/M

Recent years have witnessed a growing support for C running under CP/M, starting with Small-C and progressing to ECO-C's highly professional implementation. This support has seeded a large and continually growing body of software written for the CP/M environment.

J.E. Hendrix's Small-C

The original version of Small-C was published in *Dr. Dobb's Journal* in May 1980 by Ron Cain. In its original form, Small-C was a simple one-pass compiler that generated assembly language text for the 8080 processor. It recognized only characters, integers, and one-dimensional arrays of each. There were no Boolean operators so bitwise logical operators were used. The only loop control was the *while* statement.

The compiler was updated and extended by Jim E. Hendrix and published again in *Dr. Dobb's Journal* in December 1982. The new compiler, dubbed version 2.0 Small-C, has been extended by the addition of code optimization, data initialization, conditional compilation, the EXTERN storage class, the *for*, *do/while*, *switch*, and *goto* statements, assignment operators, Boolean operators, the one's complement operator, and assorted other features.

Since his article appeared, Jim Hendrix has published *The Small-C Handbook*, which does an excellent job of documenting the essential features of the compiler. The book, in combination with a disk of the compiler sources, would make an excellent semester course on the structure of seed compilers.

The Small-C compiler at $25, with the book $14.95, brings this tool in at the lowest price of any of the compilers we have reviewed for CP/M. The combined price tag of $39.95 is a good investment, as the compiler makes an excellent tool for doing small utility development in the CP/M environment. Be aware that it implements only a subset of the full language.

Disk one contains the compiler CC.COM, several submit files, a read.me document, and order forms for the use of people who wish to get their own copy of Small-C. The read.me document deals with a problem seen with using certain versions of M80. The only instantly apparent source file is called AR.C. We displayed the source file and discovered that it is a program archiver and the assorted .ARC files on the disk are archived sources for the system. A submit file connects the compilation, assembly, and link portions of building a program. It seems as if almost everything has been thought of. The documentation consists of a copy of *The Small-C Handbook*.

The largest plus we see to Small-C is that in combination with *The Small-C Handbook* you have a ready tool to learn about what a compiler is and how it works. The compiler does not try to deal with the entire language so it is somewhat simpler and easier to read. Modifying the compiler to extend the language is quick and easy.

The Small-C compiler is in the public domain and virtually free. What you are paying for when you purchase the package is the right to be in on any updates and fixes that come along, which is explained in the file REGISTER on the disk. It is your least expensive step into the C language. If you try it out and find it's not to your taste, you haven't made a heavy investment. If you find that you like C, then in the future you can move up to a bigger implementation.

But *be warned*, do not expect to do large-scale development in Small-C. More comprehensive versions of the language are out there. Many of the more powerful extensions of C would make some rather long and tedious Small-C programs quite a bit shorter and more concise.

Small C could run only one of the benchmarks. Lack of floating point support and one-dimensional arrays prevented the Matrix program from compiling, lack of unsigned integers kept Fib from compiling, and lack of structures stopped Deref.

Also be warned that if you do not already have an M80 compatible relocating assembler, and a linker, you are going to have to get one in order to use Small-C. Jim Hendrix does not supply one with the package. Output of the compiler is assembly language source.

Small-C is not unique in offering the entire compiler source as part of the compiler package. It was just the first. If you need to fill a special niche in your tool chest, then Small-C may fit the bill.

Software Toolworks' C/80

Software Toolworks' C/80 is a late entry into the CP/M group of compilers. Fitting between J.E. Hendrix's Small-C and The Code Works' Q/C, it is the least expensive of the C implementations offering a floating point package. The compiler comes with a configuration program and an assembler. The total package with MathPack and the compiler costs about $90. For the price it seems to be a reasonable implementation of C. However, it is not as close to K&R C as the advertising would like you to believe.

This compiler does sport some rather impressive benchmarks, taking just 26 sec to run the Sieve test. It then promptly turned around and exploded on Deref. We don't know what happened. It got lost and filled the screen with garbage when

Pascal and C Programmers

Your programs can now compile the *FirsTime*™

FirsTime is an intelligent editor that knows the rules of the language being programmed. It checks your statements as you enter them, and if it spots a mistake, it identifies it. *FirsTime* then positions the cursor over the error so you can correct it easily. *FirsTime will identify all syntax errors, undefined variables, and even statements with mismatched variable types.* In fact, any program developed with the *FirsTime* editor will compile on the first try.

Unprecedented
FirsTime has many unique features found in no other editor. These powerful capabilities include a zoom command that allows you to examine the structure of your program, automatic program formatting, and block transforms.

If you wish, you can work even faster by automatically generating program structures with a single key-stroke. This feature is especially useful to those learning a new language, or to those who often switch between different languages.

Other Features: Full screen editing, horizontal scrolling, function key menus, help screens, inserts, deletes, appends, searches, and global replacing.

Programmers enjoy using *FirsTime.* It allows them to concentrate on program logic without having to worry about coding details. Debugging is reduced dramatically, and deadlines are more easily met.

FirsTime for PASCAL	$245
FirsTime for C	$295
Microsoft PASCAL Compiler	$245
Microsoft C Compiler	$395
Demonstration disk	$25

Get an extra **$100 off** the compiler when it is purchased with ***FirsTime.***
(N.J. residents please add 6% sales tax.)

Spruce
Technology Corporation
110 Whispering Pines Drive
Lincroft, N.J. 07738
(201) 741-8188 or (201) 663-0063

Dealer enquiries welcome. Custom versions for computer manufacturers and language developers are available.

FirsTime is a trademark of Spruce Technology Corporation.

MasterCard VISA

CIRCLE 33 ON READER SERVICE CARD

attempting the pointer dereferencing. It is hoped this problem will be fixed.

The source code for the libraries is included in the package with the source for the floating point package. Software Toolworks has included at least some rudimentary verification tools with the package. It's a start at least.

We have a problem with the manual, however. The 60 pages includes a 10-page blurb on the MathPack. It covers the topic, but is reminiscent of the original CP/M user's guides that told you everything you needed to know about CP/M, if only you knew where to look. K&R C and C/80 are going to have several points of difference, so if your goal is portability, this is not the tool for you.

Although this compiler does not live up to its advertised claim of being a full C implementation, it does have a place if for no other reason than to create blazingly fast utilities. Writing a large program (over 10,000 lines) with this compiler could be a trick. The main effort at least initially would be to verify which constructs worked and which were going to go out to lunch.

The Code Works' Q/C

The second compiler we examined for CP/M was the Q/C compiler from The Code Works. This is the only other compiler we know of that sells the compiler package with source. (At least in the affordable category.)

This compiler is a very good step up from Small-C for the hobbyist. In particular, it has a respectably rich subset of the C language. It differs in that it does not support *long*, *float*, or *double* declarations, which again means that Matrix cannot be executed for this compiler. Lack of support for extended-precision numbers does not greatly reduce the value of this compiler to the experimenter. The compiler sources are included.

Q/C, like the other less expensive compilers, does not implement parameterized *#defines*. The Q/C compiler also does not support the initialization of *local*, *auto*, or *register* variables. Because Q/C is implemented as a single-pass recursive descent compiler, local variable declarations are only supported immediately after a function header; local variables within blocks and bit fields are not supported.

The preceding list is not particularly extensive for a compiler in the $100 range, particularly one distributed in source form. Educators needing a good tool with its own programming course should look long and hard at this product.

We particularly like two features available with this compiler. The first is that the compiler is available to create code for one of two processors: 8080 or Z80. The Z80 version generates only M80 (or CWA—the optional Code Works assembler) mnemonics because of optimizations for the Z80 processor. The 8080 version

Thunder Software

- **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 operating system. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a 24 page users guide, newsletters. Macro preprocessor, runs on APPLE][,][+, //e, //c. Source code for libraries is included. **Only $49.95**
- **ASSYST: The Assembler System** - A complete 6502 editor/assembler and lister for APPLE DOS 3.3. Menu driven, excellent error trapping, 24 p. users guide, demo programs, source code for all programs! Great for beginners. **Only $23.50**
- **THUNDER XREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. Source code and documentation provided. **Only $19.95**

Thunder Software POB 31501 Houston Tx 77231 713-728-5501
Include $3.00 shipping. COD, VISA and MASTERCARD accepted

CIRCLE 65 ON READER SERVICE CARD

MEMO: C Programmers

QUIT WORKING SO HARD.

These people have quit working so hard: IBM, Honeywell, Control Data, GE, Lotus, Hospitals, Universities & Government Aerospace.

THE GREENLEAF FUNCTIONS™

THE library of C FUNCTIONS that probably has just what you need . . . **TODAY!**

. . . already has what you're working to re-invent
. . . already has over 200 functions for the IBM PC, XT, AT, and compatibles
. . . already complete . . . already tested . . . on the shelf
. . . already has demo programs and source code
. . . already compatible with all popular compilers
. . . already supports all memory models, DOS 1.1, 2.0, 2.1
. . . already optimized (parts in assembler) for speed and density
. . . already in use by thousands of customers worldwide
. . . already available from stock (your dealer probably has it)
. . . It's called the **GREENLEAF FUNCTIONS.**

Sorry you didn't know this sooner? Just order a copy and then take a break — we did the hard work. Already.

THE GREENLEAF FUNCTIONS GENERAL LIBRARY: Over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and system interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you. From Greenleaf Software.

. . . **Specify compiler** when ordering. Add $7.00 each for UPS second-day air. MasterCard, VISA, check, or P.O.

◆ Compilers:
CI C86 . . . . $349
Lattice $395
Mark Williams . . . $475

◆ General Libraries $175
(Lattice, Microsoft, Mark Williams, CI C86)
◆ DeSmet C $150
◆ Comm Library $160

GREENLEAF SOFTWARE©

GREENLEAF SOFTWARE, INC.
2101 HICKORY DRIVE ◆ CARROLLTON, TX 75006 ◆ (214) 446-8641

CIRCLE 44 ON READER SERVICE CARD

oes not support Zilog mnemonics.

The second feature we liked is *trace*. When turned on, the compiler generates ode that displays the name of each function when it is executed. This sounds handy for tracing down hard to find or nvisible problems. The trace feature also causes the compiler to search for *trace* lags placed within comments and to generate the code necesary to output the text within the trace flag text area.

The manual is excellent. It starts off with an introduction by the author. The first chapter shows a sample compilation using each of the recommended assemblers. It also covers the QRESET program, which is used to change certain parameters within the compiler (size of the symbol table, macro storage, default options, etc.). This is a reasonable way of dealing with the limitations of a small machine. The sources for QRESET are part of the source library.

This manual, however, does have one failing: it has no index. Most of the material is well-organized, but certain things are hard to find, such as information on EXPAND.COM. This is not serious because of the manual's good general organization, just a nuisance.

The Q/C compiler, by Jim Colvin, has a lot to recommend it. Sources are included as part of the package. This means that self-maintenance is possible for the hobbyist, and the resource material made available to the instructor makes it a must acquisition. The price tag of close to $100 is not totally intimidating.

Some of the compiler's features (e.g, *trace*) are unusually good ideas. We would like to have this feature on some of the VAX cross compilers we use at work. In particular, the ability to embed *trace* flags and debug text for output in comments is one of those things that make you hit yourself and say "Why didn't I think of that?" Other features, such as the ability to compile code for multiple assemblers, indicate the author has run into the new compiler syndrome: being able to do the compilation but not having the assembler to finish the job. Very frustrating.

Q/C is a small machine compiler. It does what it is meant to do and it does it well. The compiler is shipped set to compile code modules with 200 symbols in its symbol table. This can be changed. Still, it indicates an inherent limitation of the system. The limit is on the total number of symbols per compilation rather than per level of scope, which means you may need to keep your modules small in order to use this compiler effectively. We can't help but wonder what the maximum limits on these settings would be.

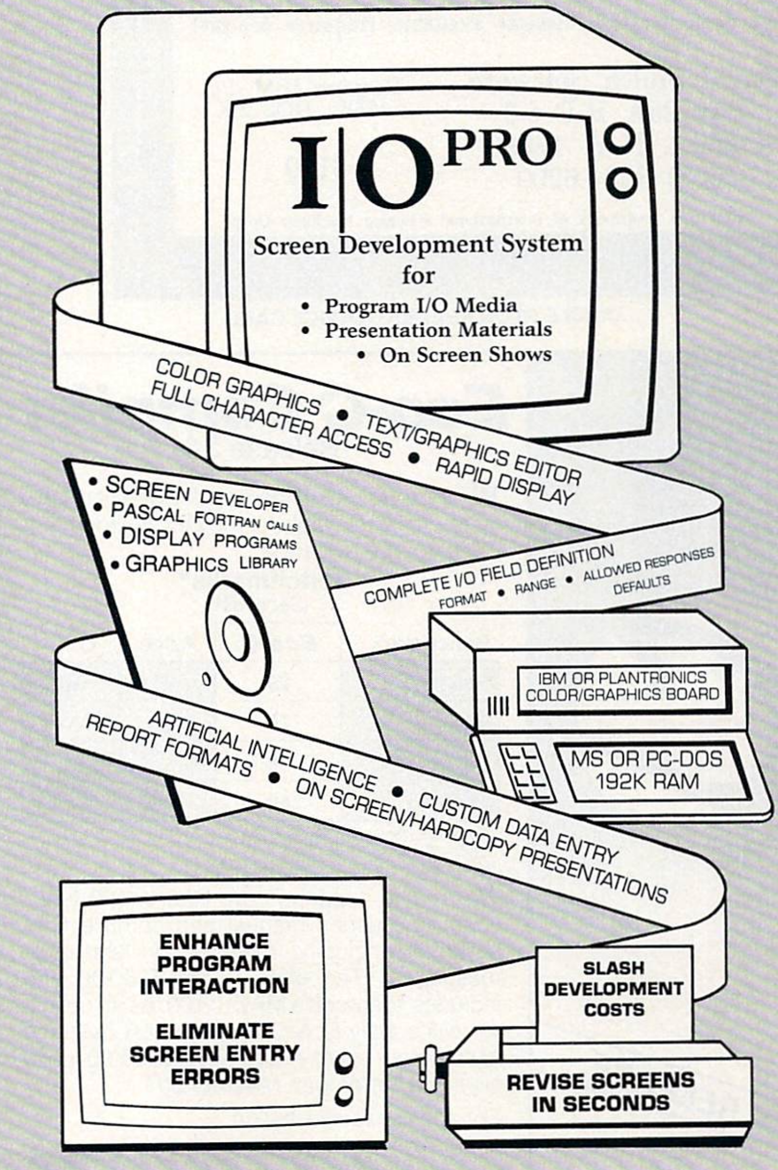
BD Software's BDS C

BD Software's BDS C has been used to write software for the CP/M operating system for about four years. In that time it has become one of the most used and respected C compilers within the CP/M community. The compiler is written in 8080 assembly language and directly produces 8080 code. This is not always a desirable feature, but in this case, because it is a blindingly fast compiler, it is forgivable. It is so fast that it can approach the convenience of an interpreter for small code fragments. BDS C accomplishes this speed by loading the entire source file from disk into memory and compiling in place.

BDS C is the only CP/M C compiler we have reviewed that has seriously examined the program verification and debugging issue and has provided tools to aid in this endeavor. This is accomplished using two methods. BDS recognizes that the CP/M environment has very effective verification tools already available. These tools are utilized by producing as an output from the compiler a standard format symbol file. This allows the the C programmer convenient access to assorted symbolic debuggers supplied by other vendors.

The compiler may also be instructed to output 8080 restart (RST) instructions at

I/O A BORE? NOT ANY MORE!



I|O PRO
Screen Development System
for
• Program I/O Media
• Presentation Materials
• On Screen Shows
COLOR GRAPHICS • TEXT/GRAPHICS EDITOR
FULL CHARACTER ACCESS • RAPID DISPLAY
• SCREEN DEVELOPER
• PASCAL FORTRAN CALLS
• DISPLAY PROGRAMS
• GRAPHICS LIBRARY
COMPLETE I/O FIELD DEFINITION
FORMAT • RANGE • ALLOWED RESPONSES
DEFAULTS
IBM OR PLANTRONICS COLOR/GRAPHICS BOARD
MS OR PC-DOS 192K RAM
ARTIFICIAL INTELLIGENCE • CUSTOM DATA ENTRY
REPORT FORMATS • ON SCREEN/HARDCOPY PRESENTATIONS
ENHANCE PROGRAM INTERACTION
ELIMINATE SCREEN ENTRY ERRORS
SLASH DEVELOPMENT COSTS
REVISE SCREENS IN SECONDS

M|E|F Environmental, Inc.
P.O. Box 26537
Austin, TX 78755 (512) 251-5543

$450 : OEM Pricing and Licenses Available
Demonstration Diskette $10
applicable to purchase

CIRCLE 15 ON READER SERVICE CARD

the beginning of each C statement. This allows CDB, BDS's symbolic C debugger, to directly control execution of the target program by setting breakpoints and allowing symbolic display and modification of variables. This is definitely a step toward reducing the drudgery of program verification.

The ability of BDS to quickly compile and integrate the resulting code into a very good symbolic debugger has gone a long way toward making BDS C one of the preferred tools in the CP/M programmer's toolchest.

The release we reviewed (version 1.50a) still suffers from a lack of real number implementation. It does, however, support a floating point function library, which is not integrated at all into the compiler. Thus, *float, double,* and *long* data types are not directly supported. Functions similar to the intrinsic *long* and *float* operations are available from the library using byte arrays.

The C Users Group (formerly the BDS C Users Group) is one of the more active software-producing users groups in the country. The *float* and *longint* package as well as the CDB symbolic debugger resulted from efforts by members of this group.

Library sources have been included to allow customization by users for non- CP/M environments. Along with these are a large number of source files. The CDB symbolic debugger is included in source form. There is a library for directed I/O, a utility for command line wild card expansion, an assembler to assemble to BDS C's own .CRL-format relocating libraries and the *float* and *longint* libraries mentioned previously. Othello (a game!), several file utilities, and finally Teledit, a large telecommunications program, are also included.

BDS C is one of the best values for an all-around useful tool. For $150 you have the fastest CP/M compiler on the market. If you thought that program turnaround was not important, then you have never had to do serious debugging. Given the difficulty of system development efforts, we want all the help we can get in building and verifying software. We wish more compilers used the intelligence and acute sense of reality that BDS C has placed in their debugging tools. Many of the mainframe-based cross development compilers have some form of hook into the compiled code, which allows a debugging system to be integrated into the software.

Because of the in-memory compilation, BDS C is best used for module development. The size of the source code body that can be compiled is limited to 48K or so in a 64K CP/M environment. In certain cases this may impose restrictions on the size and number of functions available in any one compilation.

CODE SIFTER

THE EASY TO USE

SYMBOLIC EXECUTION PROFILER

No knowledge of assembly language is needed.

Easy to use – become productive in minutes.

Online HELP at each menu complements the user manual.

Define 32 ranges in USER/DOS/BIOS with a single keystroke or enter them manually if you desire.

Use the looping feature to isolate the busy areas of your program automatically.

Symbolic statistical output can be sent to any device or file.

Adjustable sampling rate for increased accuracy.

Free demonstration diskette available. (requires deposit)

David Smith Software
Box 25A R.D.#3
Oxford, N.Y. 13830
(607) 843–6209

Requires IBM–PC
128K DOS 2.X

$119

IBM is a trademark of International Business Machines Corp.

CIRCLE 36 ON READER SERVICE CARD

Eco-C Compiler

Release 3.0

We think Rel. 3.0 of the Eco-C Compiler is the fastest full C available for the Z80 environment. Consider the evidence:

Benchmarks*

(Seconds)

Benchmark	***Eco-C***	Aztec	Q/C
Seive	29	33	40
Fib	75	125	99
Deref	19	CNC	31
Matmult	42	115	N/A

**Times courtesy of Dr. David Clark*
CNC - Could Not Compile
N/A - Does not support floating point

We've also expanded the library (120 functions), the user's manual and compile-time switches (including multiple non-fatal error messages). The price is still $250.00 and includes Microsoft's MACRO 80. As an option, we will supply Eco-C with the SLR Systems assembler - linker - librarian for $295.00 (up to six times faster than MACRO 80).

For additional information, call or write:

ECOSOFT INC. *(317) 255-6476*

6413 N. College Ave. • Indianapolis, Indiana 46220

NEW RELEASE

CIRCLE 22 ON READER SERVICE CARD

The greatest failing of the BDS C compiler has been the .CRL libraries' nonstandard format. Some sort of conversion capability that would convert between .REL format and .CRL format libraries would be an excellent addition to the system. The new CASM assembler is a good addition to the package. It just fails in many ways to link with the large body of excellent development tools available under CP/M.

Alcor C

Alcor Systems' Alcor C is one of the two CP/M implementations of C in this article that could be termed a *full* implementation. It has the most comprehensive manual of all of the implementations that we have reviewed. It is divided into six sections: beginners, editor, system, tutorial, reference, and Advanced Development Package (ADP). It contains about 470 pages of information about the use of the compiler and editor. It is well laid out with a table of contents and index for each section.

The package is divided into three main systems and the editor. The first system is the compiler and the p-code interpreter. The second system is the ADP. It contains the p-code-to-native machine conversion, the p-code optimizer, and the LinkLoader, which allows both native machine code and optimized p-code functions to be used in the same program.

The LinkLoader allows you the ability to profile the program and convert those portions requiring high efficiency to native machine code while still retaining the size efficiencies of p-code for the majority of the program. The ADP package is offered as an option to the base compiler, but for the serious development effort it is a must. The third section is Alcor's XASM multi-assembler. The assembler is not an essential portion of the system, but the documentation on the assembler contains a complete description of Alcor's low-level p-code. This is essential for doing large-scale development in this type of environment.

If your development effort is concerned with porting software, this compiler is one of the two CP/M C compilers reviewed that can do the job. Alcor C is very close to being full K&R C. Most of the differences from K&R are extensions on the base language. Alcor C has only two restrictions: the address of an array element may not be used as an initializer and external float and double variables may not be initialized.

It becomes apparent very quickly that p-code implementations have many advantages. One of them is the completeness of the implementation. The Alcor C language implementation is one

UNPARALLELED
PERFORMANCE
and
PORTABILITY
in an
ISAM PACKAGE
at an
UNBEATABLE
PRICE

c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B+ Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B+ Tree based file handlers. With c-tree™ you get:

- complete C source code written to K&R standards of portability
- high level, multi-key ISAM routines and low level B+ Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

$395 COMPLETE

Specify format:
5¼" PC-DOS 3½" Mac
8" CP/M® 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc.
c-tree and the circular disc logo are trademarks of FairCom

© 1984 FairCom

CIRCLE 29 ON READER SERVICE CARD

TOTAL CONTROL:
FORTH: FOR Z-80®, 8086, 68000, and IBM® PC
Complies with the New 83-Standard
GRAPHICS • GAMES • COMMUNICATIONS • ROBOTICS
DATA ACQUISITION • PROCESS CONTROL

- ***FORTH*** programs are instantly portable across the four most popular microprocessors.
- ***FORTH*** is interactive and conversational, but 20 times faster than BASIC.
- ***FORTH*** programs are highly structured, modular, easy to maintain.
- ***FORTH*** affords direct control over all interrupts, memory locations, and i/o ports.
- ***FORTH*** allows full access to DOS files and functions.
- ***FORTH*** application programs can be compiled into turnkey COM files and distributed with no license fee.
- ***FORTH*** Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.

Trademarks: IBM, International Business Machines Corp.; CP/M, Digital Research Inc.; PC/Forth + and PC/GEN, Laboratory Microsystems, Inc.

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities and 200 page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M® 2.2 or MP/M II, $100.00; **8080 FORTH** for CP/M 2.2 or MP/M II, $100.00; **8086 FORTH** for CP/M-86 or MS-DOS, $100.00; **PC/FORTH** for PC-DOS, CP/M-86, or CCPM, $100.00; **68000 FORTH** for CP/M-68K, $250.00.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly.

PC FORTH +	$250.00
8086 FORTH + for CP/M-86 or MS-DOS	$250.00
68000 FORTH + for CP/M-68K	$400.00

Extension Packages available include: software floating point, cross compilers, INTEL 8087 support, AMD 9511 support, advanced color graphics, custom character sets, symbolic debugger, telecommunications, cross reference utility, B-tree file manager. *Write for brochure.*

VISA

LMI

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to (213) 306-7412

MasterCard

CIRCLE 35 ON READER SERVICE CARD

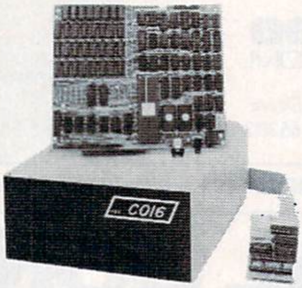
which we would not hesitate to use in a large programming effort. The professionalism presented is very apparent.

One of the elements that seems to be missing, however, is effective support for program verification. Since the compiler produces p-code that is interpreted at one level in the development, one would believe that a very good symbolic debugger could be produced. In particular, other languages are available from Alcor running the same p-codes, which would make the development of an interactive debugging tool to go with its language packages quite attractive.

The p-code implementation also allows for a large system development on a small machine. This may seem to be contradictory at first, but consider that the p-code, instruction for instruction, is much more dense than native machine code. So that once the program size has made up for the interpreter overhead, the p-code begins to be much more compact than the native code implementation.

However, for small programs and utilities, p-code loses big. Even with p-code conversion to native code, the native code compilers will win out. Interpreted languages—whether text-based or p-code-based—all have one failing: they bring one more level of complexity and potential for error into the picture.

The p-code interpreter within the Alcor package is murderously slow. The Sieve benchmark took 3.3 min to run. We aborted Fib after 5 min. This seems to be about 10 times slower than most of the native code implementations. Even with a 3-4 times improvement in speed of generated native code, it is running 2-3 times slower than the competition.



A POWERFUL 68000 DEVELOPMENT ENVIRONMENT FOR YOUR Z80 SYSTEM

CO1668 ATTACHED RESOURCE PROCESSOR

- 68000 Assembler
- C Compiler
- Forth
- Fortran 77
- Pascal
- BASIC-PLUS
- CBASIC
- APL. 68000

6 MHZ 68000 CP/M-68K 768K RAM

4 x 16081 MATH CO-PROCESSORS CPM80 RAM DISK

Develop exciting 68000 applications on your current Z80 based CPM system using powerful mini-frame like 32 bit programming languages. And then, execute them at speeds that will shame many $100K plus *mini*computer systems.

The **CO1668 ATTACHED RESOURCE PROCESSOR** offers a Z80 CPM system owner a very low cost and logical approach to 68000 development. You have already spent a small fortune on 8 bit diskette drives, terminals, printers, cards cages, power supplies, software, etc. The CO1668 will allow you to enjoy the vastly more powerful 68000 processing environment, while preserving that investment.

CO1668 ATTACHED RESOURCE PROCESSOR SPECIAL FEATURES:

- 68000 running at 6 Mhz
- 256K to 768K RAM (user partitioned between CPU and RAM Disk usage)
- Up to four 16081 math co-processors
- Real time clock, 8 level interrupt controller & proprietory I/O bus
- Available in tabletop cabinet
- Delivered w/ sources, logics, & monolithic program development software
- Easily installed on *ANY* Z80 CPM system
- CP/M-68K and DRI's *new* UNIX V7 compatible C complier (w/ floating point math) - standard feature
- *Can be used as 768K CPM80 RAM Disk*
- Optional Memory parity
- No programming or hardware design required for installation
- Optional 12 month warrantee

- PRICES START AS LOW AS $899.00 for a CO1668 with 256K RAM, CPM68K, C Complier, Sources, Prints, 200 page User Manual, Z80 Interface, and 68000 System Development Software.

For further information about this revolutionary product or our Intel 8086 Co-Processor, please send $1 [no checks please] or call:

HSC INC.

Hallock Systems Company, Inc.
262 East Main Street
Frankfort, New York 13340
(315) 895-7426

RESELLER AND OEM INQUIRIES INVITED.

CIRCLE 31 ON READER SERVICE CARD

Ecosoft's ECO-C

Ecosoft proudly states that the only incompatibilities that its Eco-C has with the full C syntax are lack of bit fields and the *#line* macro directive. The Eco-C package provides everything that is required to compile and run programs. In particular, it provides one of the best assemblers we have stumbled across, the SLR assembler by SLR Systems. The inclusion of tools of this quality does not dim the quality of Ecosoft's C implementation but complements it, creating a very good program development tool.

Eco-C is a very clean implementation of the C language. Portablility of C programs from other systems is very good. The SLR assembler selected as a companion tool to the compiler is a breath of fresh air to the assembly language programmer.

The manual for Eco-C is 115 pages long in an off-size format binder. For the large-scale project, you should see if you can order several manuals at extra cost. It is very concise and professionally done, concentrating on the essentials of operating the compiler, creating libraries, the assembler interface, and Ecosoft's version of the standard library interface. The compiler does not include any sources for the STDIO but does state that Ecosoft will make them available for a simple copying charge and release statement.

Included in the same binder is the manual for the SLR assembler. Approximately the same length as the compiler manual, it quickly illustrates the use of the assembler, linker, and librarian included in the package. Just as an example, the assembler can be configured from the command line to output either a .COM file, a .HEX file or one of two separate .REL file formats. We have not even discussed how much faster it executes than M80. Too bad we are not reviewing assemblers!

At $250, the Eco-C package is out of the range of most individual programmers. Even so, with the inclusion of the SLR assembler for $45 extra, it is a good investment. The compiler and assembler are good for a large range of software development, from the small utility development effort to massive multi-programmer efforts. This is the only package we reviewed that we would be willing to call a professional tool. The compiler options, although not simple, were well-documented. The benchmarks were not the fastest but were in the ballpark, making this a fine tool for all-around use.

The compiler is such a nice tool that it is too bad program verification interfaces consist entirely of using the .SYM output file at the assembly language level. There should be some way of hooking a symbolic debugger to the final program, either as a compiler option to output restart instructions or hooks to allow for post processing the assembler output (i.e., output line numbers and line boundary markers as a compiler option).

OS-9 C COMPILERS

Microware C

Microware offers one of the few C compilers for the OS-9 operating system and the 6809 processor. That's not too much of a surprise because OS-9 is a Microware operating system. In case you haven't run into OS-9 before, it is a multi-user and multitasking operating system for the Motorola MC6809 8/16-bit microprocessor. OS-9 is blessed with UNIX-like tree directories and can support over 1MB of system memory with memory management. It is therefore a very comfortable environment for the C language.

Though it is very close to a full C implementation, Microware C differs somewhat from K&R C. Bit fields are not supported. Constant expressions for initializers may include arithmetic operators only if the operands in the expression are of type *int* or *char*. The *#if* < *constant expression* > is not supported although *#ifdef*, *#ifndef*, *#else* and *#endif* are. Macro definitions and strings must reside on a single line of source code. Finally, the \n escape sequence is output as a carriage return, which is not interpreted as an "end of line" character in most systems but is exactly what OS-9 requires.

On the positive side, Microware has added some interesting enhancements. A new storage class, *Direct*, has been added to take advantage of the 6809's direct page register. Though use of this feature will make a program non-portable, it can be used when program execution speed has the highest priority.

In addition, Microware's C supports in-line assembly code with the *#asm* and *#endasm* preprocessor commands. It is possible to output a linefeed using the \l line feed escape sequence extension.

The Microware level II compiler comprises six separate programs. An execution manager, called CC2, is supplied as one of these programs. CC2 invokes the other five programs, simplifying the programmer's interface. The compilation sequence involves the preprocessor, compiler, optimizer, assembler, and linker. Also included in the package is an extensive library of header files that facilitate operating system calls, a Microware BASIC09 interface, and several other useful functions.

Introl C

Introl offers the other C compiler for the OS-9 operating system reviewed in this article. It is available for the Flex and Uniflex operating systems as well. Introl C is almost a full K&R implementation, lacking bit fields, *double float* variables and the *#line* and *#if* preprocessor directives.

Introl's compiler has some enhancements. Comment nesting is permitted, allowing the programmer to comment out a section of code by bracketing the code segment. Separate name spaces for each union and structure are supported so that identical names can be used in different *struct* and *union* declarations. Symbols can be up to 90 characters in length.

For $425 Introl supplies the compiler, a relocating assembler, linker, a library and a librarian. This comprises a nearly complete OS-9 language development system, lacking only an editor. For the hardware developer, a library of modules is available for generating stand-alone, ROMable code. Introl's documentation is superb.

One amenity Introl C lacks, however, is a module to invoke the various compiler phases automatically. With preprocessor, parse, code generation, optimizer, assembler and link phases all required to generate an executable file, such a utility would be most welcome. Introl's C compiled more slowly than Microware's compiler and generated larger executable files. However, this is offset by the faster execution times of Introl C's compiled programs.

CP/M 86 C COMPILERS

The last MS-DOS C compiler we reviewed is from Digital Research, which submitted both versions for review. It is similar to the CP/M 86 version, so we discuss both in this section.

CP/M 86 has not done well in the IBM

Realia COBOL.
Numbers speak louder than words.

Compilation Speed (minutes:seconds)

Lines in Program	Realia COBOL	mbp COBOL	Level II COBOL	R-M COBOL	Microsoft COBOL
1,000	:51	8:33	3:42	5:05	5:11
5,000	3:30	48:07	16:58	*	45:26

*Could not successfully compile the program.

Execution Time Ratio (Gibson Mix; calculated S-Profile)

Realia COBOL	mbp COBOL	Level II COBOL	R-M COBOL	Microsoft COBOL
1.0	3.6	14.7	21.6	22.3

All benchmark tests were performed on an IBM PC-XT with 192KB of memory. IBM PC-XT is a registered trademark of International Business Machines Corporation; mbp COBOL, of mbp Software and System Technology; Level II COBOL, of Micro Focus; R-M COBOL, of Ryan-McFarland; and Microsoft COBOL, of Microsoft.

Sieve of Eratosthenes

0.818 seconds per iteration

+ IBM VS COBOL compatibility.

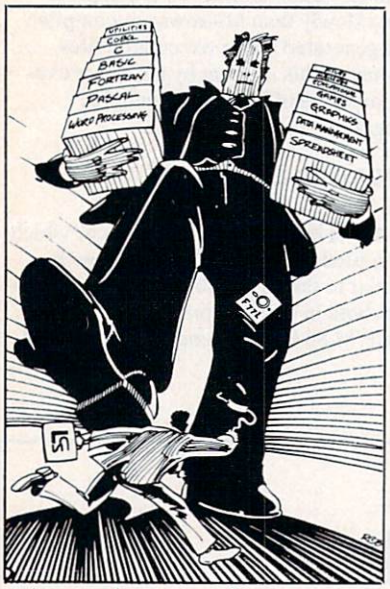
$995

REALIA inc.

10 South Riverside Plaza
Chicago, Illinois 60606
(312) 346-0642

CIRCLE 76 ON READER SERVICE CARD

F77L, THE FORTRAN THAT CHALLENGES THE BIG SOFTWARE COMPANIES.



When you have one software product to sell, you had better make it count. F77L, our complete implementation of the ANSI FORTRAN 77 Standard for the IBM PC, is for programmers who buy a language system based on features and performance.

By specializing in FORTRAN, we see ourselves as different from our competition. Rather than be jacks of all languages, LCS prefers to be experts in FORTRAN. Our competitors, on the other hand, tend to be large and ferocious software firms that offer computer users an assortment of products. At LCS, we specialize in FORTRAN. We have been successfully implementing FORTRAN language systems for over 15 years. We may not have the name recognition or the advertising budgets of the big firms, but we do have what matters most to you—a great product.

Here are a few of the many reasons to buy F77L:

- *Full* FORTRAN 77 Language with popular extensions.
- Compile time: multiples faster than any Goliaths'. (100's statements/min.)
- User oriented interface.
- Numerous, specific English level diagnostics displayed during compilation.
- Command Line compiler options.
- Execution error traceback: program unit line number.
- Selective protection for constants, bounds, interfaces.
- Standard or Free Format source files.
- Lattice C compatibility.
- Easy to follow manual includes appendices on interfaces to Lattice C and Assembly language.
- 30-day-money-back guarantee and ongoing user support.

If you're tired of betting on the software Goliaths and losing, call LCS, the FORTRAN specialist.

$477 for complete package: one 5¼" floppy and manual. Visa/MC Multiple copy discounts. *Requires: 256K/8087.

To order or for more information:

Lahey Computer Systems, Inc.
904 Silver Spur Road, Suite 417
Rolling Hills Estates, CA 90274
213/541-1200

Serving the FORTRAN community since 1969

IBM is a trademark of IBM corporation
Lattice C is a trademark of Lattice, Inc.

CIRCLE 62 ON READER SERVICE CARD

PC segment of the 16-bit market but has fared better with the other computers using the 8086 processor. Since CP/M 86 is a Digital Research operating system, it is of no surprise that the one CP/M 86 C compiler tested is also from Digital Research, though many of the MS-DOS C compilers are also available in CP/M 86 versions.

Digital Research C

We had a bit of trouble coaxing the CP/M 86 version of Digital Research C to run on our IBM PC. This was probably not the compiler's fault. We had an old copy of CP/M 86 (version 1.0), and the compiler simply returned to the operating system before completing the compilation. Version 1.1 of CP/M 86 cleared this up, and we were off and running.

Digital Research C supports the small and big 8088 memory models. Memory model selection as well as 8087 support are specified at compile time by command line options.

One very nice feature of the Digital Research C compiler documentation is the inclusion of a copy of K&R with the manual. Digital Research made it quite clear how to configure a working disk with compiler, linker, and library. We had no problem creating one.

Source code was another story. Our benchmarks had all been written using a text editor under MS-DOS, and we had no editor for CP/M 86. Copies of CP/M 86 compatible editors are few and far between in our area, and we turned a bit green when we considered using the CP/M 86 editor ED to create the files. Fortunately we have a very handy utility program called Xenocopy Plus from Vertex Systems, which promptly transferred the four benchmark source files to a CP/M 86 disk.

Sieve compiled and ran with no problem. Fib required that we add *#include <portab.h>* so that the *void* typing would be recognized. Deref and Matrix also compiled and ran fine.

An interesting aspect of the compilation is that the compiler places information in the object file so that the linker knows which libraries to use. No library parameters are in the link syntax. The compiler was not smart enough to place the object file on the same disk as the source file. It required we specify we wanted this by using the -z compiler option. However, the linker knew that we wanted the executable file on the same disk as the object file. Consistency once again is desirable but lacking.

Digital Research also submitted for review the MS-DOS version of its C compiler. Its benchmark performance is included in both the MS-DOS and CP/M 86 benchmark tables for comparisons. Two very interesting conclusions are that Digital Research's compiler generates very fast code, consistently showing one of the best execution times in the MS-DOS category.

The other point of note is that the MS-DOS version of the compiler generates programs more quickly than the CP/M 86 version. Though not shown in the tables, we checked the times required to compile and link separately. The MS-DOS versions of both the compiler and linker are faster than their CP/M 86 counterparts. We suspect that this is actually better performance on the part of MS-DOS during floppy accesses.

TRS-80 C COMPILERS

Three C compilers are available for the TRS-80 Z80 computers: LC, Alcor C, and Aztec C. Because other machine versions of the Alcor and Aztec C compilers have already been covered, we will focus here on MiSosys's LC compiler.

LC is based on Small-C and, like Small-C, is not a full implementation of the language. However, LC provides more of C's features than does Small-C. The *switch* and *goto* statements are included, greatly simplifying command parsers and similar routines. The omissions (floating point capability, *long* integers, *unions*, and *structs*) are significant only if you try to use it as a number cruncher.

LC first appeared on the scene in December 1982 and has undergone only two significant changes since then. The first was a general bug-fix shortly after the package was released, which brought the version number to 1.1; the second actually did not affect the compiler but rather greatly extended the supplied libraries. This, the current version, is 1.2. This review was done using version 1.1.

When you buy LC (for $150), you get two disks (and a large 3-ring binder filled with an excellent user manual). One disk contains the LC compiler itself together with the libraries, an /ASM skeleton, and a /JCL file. The other contains EDAS, a full-featured macro assembler (which can also be purchased separately). The LC strategy is to compile the C source code into an assembly-language file, which is then assembled by EDAS. EDAS produces executable code as its output; no linking step is required.

On TRS-80 model 1 or 3, LC programs require the LDOS operating system. No other DOS provides the capability for I/O redirection. On model 4, LC programs will run under all versions of TRSDOS 6. Incidentally, by using the appropriate /ASM skeleton and libraries, LC programs can be cross compiled between models 1, 3, and 4. Any model 1/3 LC program will run on either model interchangeably; model 4 programs interface to the hardware differently but can be compiled by the model 3 compiler.

The assembly-language output from LC

Scroll & Recall™

Screen and Keyboard Enhancement
for the IBM - PC, XT and Compatibles

Allows you to conveniently scroll back through data that has gone off the top of your display screen.

Allows you to easily recall and edit your previously entered DOS commands and data lines.

Very easy to use, fully documented. Compatible with all versions of DOS, monochrome & graphic displays.

***$69** - Visa, M/C, Check, COD, POs*
Phone orders accepted

Make Your Work Easier!

To Order or to Receive Additional Information, Write or Call:

Opt-Tech Data Processing
P.O. Box 2167 • Humble, Texas 77347
(713) 454-7428
Dealer Inquiries Welcome

CIRCLE 67 ON READER SERVICE CARD

SUPER FORTH 64®

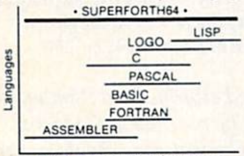
TOTAL CONTROL OVER YOUR COMMODORE-64™
USING ONLY *WORDS*
MAKING PROGRAMMING FAST, FUN AND EASY!

MORE THAN JUST A LANGUAGE...
A complete, fully-integrated program development system.
Home Use, Fast Games, Graphics, Data Acquisition, Business
Real Time Process Control, Communications, Robotics, Scientific, Artificial Intelligence

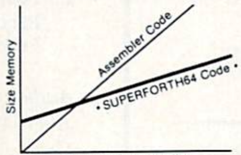
A Powerful Superset of MVPFORTH/FORTH 79 + Ext. for the beginner or professional

- 20 to 600 x faster than Basic
- 1/4 x the programming time
- Easy full control of all sound, hi res. graphics, color, sprite, plotting line & circle
- Controllable SPLIT-SCREEN Display
- Includes interactive interpreter & compiler
- Forth virtual memory
- Full cursor Screen Editor
- Provision for application program distribution without licensing
- FORTH equivalent Kernal Routines
- Conditional Macro Assembler
- Meets all Forth 79 standards+
- Source screens provided
- Compatible with the book "Starting Forth" by Leo Brodie
- Access to all I/O ports RS232, IEEE, including memory & interrupts
- ROMABLE code generator
- MUSIC-EDITOR
- SPRITE-EDITOR
- Access all C-64 peripherals including 4040 drive
- Single disk drive backup utility
- Disk & Cassette based. Disk included
- Full disk usage — 680 Sectors
- Supports all Commodore file types and Forth Virtual disk
- Access to 20K RAM underneath ROM areas
- Vectored kernal words
- TRACE facility
- DECOMPILER facility
- Full String Handling
- ASCII error messages
- FLOATING POINT MATH SIN/COS & SQRT
- Conversational user defined Commands
- Tutorial examples provided, in extensive manual
- INTERRUPT routines provide easy control of hardware timers, alarms and devices
- USER Support

SUPER FORTH 64® *is more powerful than most other computer languages!*



SUPER FORTH 64® *compiled code becomes more compact than even assembly code!*



A SUPERIOR PRODUCT
in every way! At a low
price of only
$96

Call:
(415) 651-3160
PARSEC RESEARCH
Drawer 1776, Fremont, CA 94538

Take this ad to your local dealer, or B. Dalton Bookstore. Phone orders also accepted. Immediate delivery! Dealer inquiries invited. CA residents must include tax.

or C.O.D.

® PARSEC RESEARCH (Established 1976)
Commodore 64 & VIC-20 TM of Commodore

CIRCLE 49 ON READER SERVICE CARD

MicroMotion

MasterFORTH

It's here — the next generation of MicroMotion Forth.

- **Available for the APPLE II's, IBM PC, Macintosh & CP/M 2.x.**
- Meets all provisions, extensions and experimental proposals of the FORTH-83 International Standard.
- Uses the host operating system.
- Built-in micro-assembler with numeric local labels.
- A full screen editor is provided which includes 16 x 64 format, can push & pop more than one line, user definable controls, upper/lower case keyboard entry, A COPY utility moves screens within & between lines, line stack, redefinable control keys, and search & replace commands.
- Includes all file primitives described in Kernigan and Plauger's Software Tools.
- The input and output streams are fully redirectable.
- The editor, assembler and screen copy utilities are provided as relocatable object modules. They are brought into the dictionary on demand and may be released with a single command.
- Many key nucleus commands are vectored. Error handling, number parsing, keyboard translation and so on can be redefined as needed by user programs. They are automatically returned to their previous definitions when the program is forgotten.
- The string-handling package is the finest and most complete available.
- A listing of the nucleus is provided as part of the documentation.
- The language implementation exactly matches the one described in MASTERING FORTH, by Anderson & Tracy. This 200 page tutorial and reference manual is included with MasterFORTH.
- Floating Point & HIRES options available.
- MasterFORTH – $100.00 APPLE & CP/M; $125.00 Macintosh & IBM PC. Floating Point & HIRES – $40.00 each.
- Publications
  - MASTERING FORTH - $18.00
  - 83 International Standard – $15.00
  - FORTH-83 Source Listing 6502, Z-80, 8086 - $20. ea.

Contact:
MicroMotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, CA 90025
(213) 821-4340

CIRCLE 73 ON READER SERVICE CARD

can be modified or optimized as desired. No optimizer is included with the package.

Data types handled by LC include *char*, *int*, *unsigned int*, and pointers to any of these. Arrays are permitted but are limited to only one dimension. Pointer arithmetic is in exact accord with K&R as are most of the library functions. The lack of floating point capability is addressed by a special library that provides interfaces to the floating point code in the TRS-80's ROM. This makes number crunching possible, but such programs are not portable.

Since LC is a subset, only the Sieve and integer-sort benchmarks are applicable to it. Table 5 (page 102) presents the results (measurements were made on an early-production model 4 operating in model 3 mode at a clock speed of 2.2 MHz).

The absence of *structs* looks to us to be only a minor inconvenience (the other omissions have posed no problems in my applications). Programs created with it include a virtual-memory full-screen editor, a communications program including multiple file-transfer protocols, and many small utilities. One of the best features of LC is the ease with which its libraries may be modified; possibly the poorest parts of it are the lack of optimization, the failure to perform constant expression evaluation at compile time, and the absence of parameterized *#define* commands.

Whitesmiths footnote

Unfortunately, we were not able to obtain for review a current version of the Whitesmiths C compiler. By the time you read this review, Whitesmiths should be shipping version 3.0, which claims to improve some of the problems we encountered when we analyzed an older version (version 2.2) of the compiler.

If you're considering the puchase of this older version of the Whitesmith's compiler, you should be aware of the following concerns: installation is potentially difficult because files are automatically erased during a relatively long link session. You'll need a hard disk to run the supplied DOS 2.0 batch files. Also, the two manuals supplied do not cover certain MS-DOS implementation issues very well (e.g., no installation procedure for floppy-based systems running DOS 2.0 or 3.0), and the benchmarks used in this review had to be modified for them to compile correctly, due to the lack of automatic initializers. However, if you do cross development work with minicomputers and MS-DOS equipment, this version of Whitesmiths may be the right choice.

That's it!

A package is available for just about any requirement. Most of the packages are good, solid products. We have tried to point out the few that seemed less than adequate. Since everyone has different needs, it is difficult to make a hard recommendation.

The C programming language continues to evolve. Originating as a minicomputer systems programming language, it is now beginning to appear as a serious contender in the microcomputer arena. Is C a reality on microcomputers? The answer to that question is undoubtedly "Yes!"

References

1. Clark, David D. "Two More Versions of C for CP/M." *BYTE* (May 1984): 246.
2. Pournelle, Jerry. "A BASIC and Pascal Benchmark, Elegance, Apologies and FORTH." *BYTE* (Oct. 1982): 254.

Authors Steve Leibson, Fred Pfahler, and Jim Reed can be reached at Cadnetix Corp., 5757 Central Ave., Boulder, Colo. 80301.

Dr. Dobbs says,
"WE HAVE GOOD NEWS...

"WINDOWS FOR C can offer you a convenient, reliable means of implementing windows in your text handling programs
"..video output is fast and clean...
"..documented in well-written and readable English...
"..has obviously been well thought out and very cleanly executed.
"WINDOWS FOR C is a professionally-oriented set of programming tools...that we can recommend."

—Ian Ashdown (*Dr. Dobb's Journal, November 1984*)

MORE THAN A WINDOW DISPLAY SYSTEM, WINDOWS FOR C is a video toolkit for all screen management tasks: pop-up menus and help files, status line, keyboard interpreter, word wrap, auto scroll, highlighting, color control, overlay and restore, plus a library of over 50 building block subroutines.

EASY TO LEARN. EASY TO USE. Once you've tried WINDOWS FOR C, you'll wonder how you ever managed without it.

Full support for IBM PC/XT/AT and compatibles, plus interfaces for non-IBM computers; Lattice C, CI-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS).

The Good News...
WINDOWS FOR C
ADVANCED SCREEN MANAGEMENT MADE EASY

A Professional Software Tool From
CREATIVE SOLUTIONS • 802 • 848-7738
21 Elm Ave., Box L2 • Richford, VT 05476

WINDOWS FOR C (specify compiler and version) **$195**
Demo disk and manual (applies toward purchase) $30

MasterCard & Visa Accepted
Shipping $2.50
VT residents add 4% tax

CIRCLE 28 ON READER SERVICE CARD

ADVANTAGE #1

At Programmer's Connection, we deliver the latest versions of products and keep you informed about upcoming "new releases" so that you can make the best purchasing decision. We care about the value you receive for your dollar, and that means getting the newest version of a product with the latest enhancements. When you are expecting a new release, Programmer's Connection delivers.

PC
NEW RELEASE
NEW RELEASE
M^cGINNIS 84

Discover the advantages of buying from Programmer's Connection:

1. We offer the latest version of a product.
2. Most popular products are in stock ready to be shipped.
3. Receive same manufacturer's support as if buying direct.
4. Experienced professional programmers are on staff.
5. Select from over 200 of the best software products available.
6. Knowledgeable and courteous sales staff.
7. Significant discounts off of retail prices.
8. No extra charge on prepaid orders, including major credit cards.
9. Reasonable charges for shipping and handling.
10. Toll free service from Canada and the Continental U.S.

Programmer's Development Tools:

C LANGUAGE:

	List	Ours
Computer Innovations C-86	395	309
DeSmet C Compiler with Debugger	159	145
Lattice C Compiler	500	295
Mark Williams C Compiler w/Source Debugger	500	449
Wizard C	450	409
Xenix Development System by SCO	1350	1099

C Interpreter - RUN/C

An excellent way to learn the C language. In addition, you can use this system to develop and debug C programs before compilation.

Our price **$129**, List price $150.

OTHER LANGUAGES:

	List	Ours
8088 Assembler w/Z-80 Translator 2500 AD	100	89
BetterBASIC by Summit Software .... *Sale!*	200	149
Janus/ADA + Tools by R&R	700	499
Modula-2/86 by Logitech	495	439
Professional BASIC by Morgan Computing	95	89

STSC APL*Plus/PC

New version 4.0 of this powerful APL development system is now available. Call us for product details.

Sale price **$469**, List price $595.

C UTILITIES:

	List	Ours
Btrieve by SoftCraft	250	194
Communications Library by Greenleaf	160	124
C Power Paks from Software Horizons	CALL	CALL
C-Tree by Faircom	395	359
C To dBase by Computer Innovations	150	139
C Utility Library by Essential Software	149	119
ESP for C by Bellesoft	349	319
GraphiC by Scientific Endeavors	195	169
Greenleaf C Functions Library	175	129
Halo *Color Graphics*	200	125

C UTILITIES:

	List	Ours
Phact by Phact Associates	395	359
Windows For C by Creative Solutions	195	159

PANEL Screen Designer and Editor by Roundhill

Interactively create data entry screens. Interface directly with C & other languages.

Sale price **$199**, List price $295.

OTHER UTILITIES:

	List	Ours
Dr. Halo by Lifeboat	95	79
Periscope *debugger* by Data Base Decisions	295	269
Pfix-86 Plus by Phoenix Software	395	299
Plink-86 *Overlay Linkage Editor*	395	299
Pmate *text editor* by Phoenix Software	225	159
Polytron Products	CALL	CALL
Profiler by DWB & Associates *New Low Price!*	125	99
Screen Sculptor by Software Bottling	125	109

CodeSmith-86 Symbolic Debugger by Visual Age

The New Version 1.9 now supports In-Line Code Assembly.
Save $26 off Manufacturers List Price of $145.

Special Price **$119**

Prices are subject to change without notice.

Call for our Catalog consisting of 200 + Programmer's Development Tools Exclusively for IBM PC's and Compatibles.

VISA — Account is charged when order is shipped. — MasterCard

1-800-336-1166

In Canada:
1-800-225-1166

Programmer's Connection
136 Sunnyside Street
Hartville, Ohio 44632
(216) 877-3781 (In Ohio)

"Programmers Serving Programmers"

CIRCLE 84 ON READER SERVICE CARD

Announcing a

TOTAL PARSER GENERATOR

<GOAL> :: = <RAPID> <COMPILER> <DESIGN>

SLICE YOUR COMPILER DEVELOPMENT TIME

An LR(1) parser generator and several sample compilers, all in Pascal for your microcomputer.

- Generates parser, lexical analyzer and skeleton semantics
- Universal, error recovery system
- Adaptable to other languages and computers
- Interactive debugging support
- Thorough documentation
- TURBO PASCAL™ INCLUDED FREE OF CHARGE
- Includes mini-Pascal compiler, assembler, simulator in SOURCE

SPECIAL INTRODUCTORY OFFER $ 995

QPARSER™ runs on IBM PC/DOS in Turbo Pascal. Parser generator in object form; all else in source. QPARSER takes a grammar and generates a correct, complete, high-performance compiler with skeleton semantics in Pascal source. Easy to add full semantics for YOUR application. Excellent for industrial and academic use. An accompanying textbook (SRA publishers) available in 1985. Training can be arranged. Demo disk available for $50.

Educational and quantity discounts available. Check, money order, Mastercard, Visa. California residents add 6.5% sales tax.

WRITE OR CALL FOR FREE BROCHURE.
Technical details: call 408/255-5574. Immediate delivery. CALL TODAY!

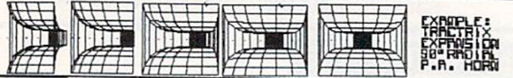
QCAD
SYSTEMS, INC.
1164 Hyde Ave., San Jose, CA 95129

TOLL FREE: 800-538-9787
(California residents call 408/255-5574)

TM Turbo Pascal is a registered trademark of Borland International.

CIRCLE 23 ON READER SERVICE CARD

What else could you do, if you had a
3-D GRAPHICS LANGUAGE to harness the
computational power of your computer?


EXAMPLE:
TRACTRIX
EXPANSION
90° RADIAL
P.A. HORN

E.G. is a new programming language that
provides most of the computing power of
a dedicated CAD SYSTEM at an incredible
price: less than one cent on the dollar

E.G. is a complete graphics language:
* automate the generation of drawings
* design math intensive products
* unwrap & template curved surfaces
* output ascii files to drive n.c.tools
* accuracy exceeds 1 part in 32,000

43 commands control looping, branching
disc access, internal data structures,
subroutines and output devices.

E.G. is also a powerful post-processor:
* plot your numerical data in 3-D
* virtually unlimited macro capability
* write "turn-key" applications
* 1000 data points, 1000 lines per file
* optional "best fit" plot algorithm

Easy to use, rather than easy to learn

E.G. provides the programmer with every
mathematical operation & function found
in BASIC + powerful, proprietary shape
generating and manipulating commands.

EUCLID GRAPHICS
SYSTEM: developed at MANUAL $ 25
64K APPLE SEVEN SEAS SOFTWARE COMPLETE $200
D.O.S 3.3 35 Cape George Wye free updates
Port Towensend, Wa. for one year
98368
(206) 385-3771

signed nondisclosure agreement required

CIRCLE 4 ON READER SERVICE CARD

C Source Code

RED

Full Screen Text Editor

IBM PC, Kaypro, CP/M 80 and CP/M 68K systems.

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
- RED is easy to use for writers or programmers. RED's commands are in plain English.
- RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.
- RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
- RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.

RED: $95

Manual: $10

edward k. ream

Call or write today for
for more information:
Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

To order:

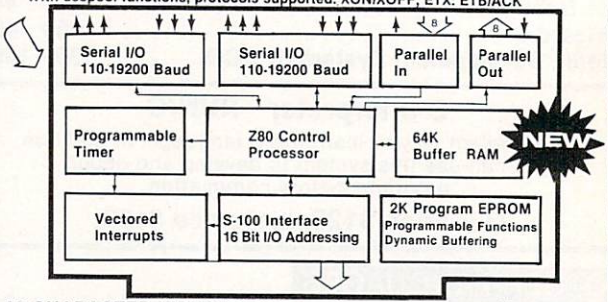
Either the BDS C compiler or the Aztec CII compiler is required for CP/M80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5¼ inch disk).

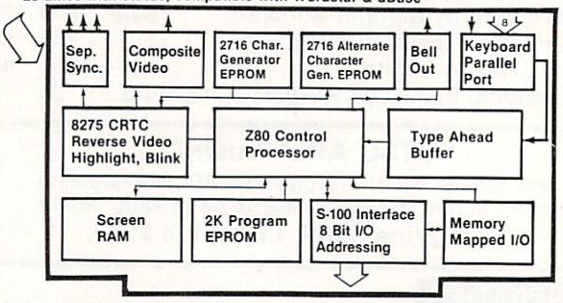
Send a check or money order for $95 ($105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.

BUFFERED I/O BOARD Introductory Price * $59.95
With despool functions, protocols supported: XON/XOFF, ETX: ETB/ACK


Serial I/O 110-19200 Baud
Serial I/O 110-19200 Baud
Parallel In
Parallel Out
8
8
Programmable Timer
Z80 Control Processor
64K Buffer RAM
NEW
Vectored Interrupts
S-100 Interface 16 Bit I/O Addressing
2K Program EPROM Programmable Functions Dynamic Buffering

80 CHARACTER VIDEO BOARD * $49.50
25 Lines with status, compatible with Wordstar & dBase


Sep. Sync.
Composite Video
2716 Char. Generator EPROM
2716 Alternate Character Gen. EPROM
Bell Out
Keyboard Parallel Port
8
8275 CRTC Reverse Video Highlight, Blink
Z80 Control Processor
Type Ahead Buffer
Screen RAM
2K Program EPROM
S-100 Interface 8 Bit I/O Addressing
Memory Mapped I/O

Includes Bareboard, Heatsink & Documentation Call or write for more information.

Simpliway Products Co.
P.O. Box 601
Hoffman Estates, IL. 60195
(312) 359-7337

SPC

VISA

MasterCard

OEM dealer pricing available, $3.00 S/H, IL. Res. add 7% tax
dBase™ - of Ashton - Tate Corp. — Wordstar™ - of Micropro Int'l. Corp

CIRCLE 70 ON READER SERVICE CARD

Basic data types

	Manufacturer	Char	Unsigned char	Int	Unsigned int	Short	Unsigned short	Long	Unsigned long	Float	Double	Pointer	Enumerated	Void	Struct	Union	Array	Bit field	Strings
MS-DOS/PC-DOS compilers	Lattice	yes	no	yes	yes	yes	no	yes	no	yes	yes	yes	no	no	yes	yes	yes	yes	yes
	Microsoft (2.03)	yes	no	yes	yes	yes	no	yes	no	yes	yes	yes	no	no	yes	yes	yes	yes	yes
	Microsoft (3.0)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	Mark Williams	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	Rational Systems	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	no	yes
	Wizard Systems	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	Computer Innovations	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	no	yes	yes	yes	no	yes
	Supersoft	yes	yes	yes	yes	yes	no	no	no	no	no	yes	?	yes	yes	no	yes	no	yes
	Manx Software Systems	yes	yes	yes	no	yes	no	yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes
	C-Systems	yes	no	yes	yes	yes	no	yes	no	yes	yes	yes	no	no	yes	yes	yes	yes	yes
	C Ware	yes	no	yes	yes	no	no	yes	no	yes	yes	yes	no	yes	yes	yes	yes	no	yes
	Datalight	yes	no	yes	no	no	no	no	no	no	no	yes	no	no	no	no	*	no	yes
CP/M compilers	Ecosoft	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	Alcor	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	BDS	yes	yes	yes	yes	yes	yes	**	**	**	**	yes	yes	yes	yes	yes	yes	yes	yes
	Code Works	yes	yes	yes	yes	yes	yes	no	no	no	no	yes	yes	yes	yes	yes	yes	no	yes
	Software Toolworks	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	yes
	Hendrix	yes	no	yes	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no
OS-9	Microware	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no		yes	yes	yes	no	yes
	Introl	yes (unsigned)	(default)	yes	yes	yes	yes	yes	yes	yes	no	yes	no		yes	yes	yes	no	yes
CP/M 86	Digital Research	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	?	yes	yes	yes	yes	no	yes

*One dimensional. **Is a library function.

Table 3.

Library functions

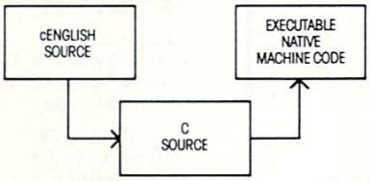
	Manufacturer	I/O	File management	String	Memory management	Arithmetic	Sort	Program control	Date/time	OS calls	Trigonometric/transcendental	Library source included
MS-DOS/PC-DOS compilers	Lattice	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	no
	Microsoft 2.0	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	no
	Microsoft 3.0	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no
	Mark Williams	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	no
	Rational Systems	yes	yes	yes	yes	yes	no	no	yes	yes	no	no
	Wizard Systems	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	Computer Innovations	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes
	Supersoft	yes	yes	yes	yes	yes	no	yes	no	yes	yes	yes
	Manx Software Systems	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	no
	C-Systems	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no
	C Ware	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no
	Datalight	yes	yes	yes	yes	yes	no	no	no	no	no	yes
CP/M compilers	Ecosoft	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	$10
	Alcor	yes	yes	yes	yes	yes	no	yes	no	yes	yes	yes
	BDS	yes	yes	yes	yes	yes	no	yes	no	yes	external	yes
	Code Works	yes	yes	yes	yes	yes	no	yes	no	yes	no	yes
	Software Toolworks	yes	yes	yes	yes	yes	no	yes	no	yes	yes	yes
	Hendrix	yes	yes	yes	yes	yes	no	yes	no	yes	no	yes
OS-9	Microware	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	no
	Introl	yes	yes	yes	yes	yes	no	yes	yes	?	no	no
CP/M 86	Digital Research	yes	yes	yes	yes	yes	yes	yes	no	no	yes	no

Table 4.

c ENGLISH.™
The C Generation Language.

What is cENGLISH? cENGLISH is a comprehensive fourth generation procedural language based on dBASE II™ syntax. It is portable to a wide range of micros and minis. The language features user-transparent interfaces to a wide range of popular C compilers, operating systems, and data base managers.

How is portability achieved? cENGLISH through its compiler interface translates cENGLISH into documented C source and uses a host C compiler to produce native machine code.



C source can be embedded in cENGLISH source.

Differences in the operating system and data base manager are handled by the runtime libraries.

The result is that cENGLISH source can be compiled without modification on any micro or mini configuration supporting cENGLISH.

What about performance? cENGLISH executes FAST, just like any compiled C program.

How easy is cENGLISH to use? While cENGLISH is a powerful high level language that can accommodate complex software development, it remains simple and straightforward to use.

Call or write for availability of cENGLISH for the following configurations—

Compilers:
  Standard O/S compilers: Lattice C™ for MS/DOS™
Operating Systems:
  UNIX,™ UNIX-like, MS/DOS,™ Coherent,™ VMS™
Data Base Managers:
  C-ISAM™ and INFORMIX,™ UNIFY,™ ORACLE,™ PHACT,™ Logix™
Foreign Language Versions:
  German, French, Spanish

Attention MS/DOS users. Demo version and special introductory offer available for IBM PC,™ XT,™ AT,™ and other MS/DOS systems. Requirements: 256K, hard disk or two floppy disk drives, and MS/DOS 2.1 or higher.

Attention dBASE II and dBASE III users. dBASE II to cENGLISH Converter now available; dBASE III Converter available later this quarter. Converted code is portable to micros or minis and executes as fast as original cENGLISH source.

dBASE II and dBASE III are trademarks of Ashton-Tate. Lattice is a trademark of Lattice, Inc. UNIX is a trademark of Bell Laboratories. MS/DOS is a trademark of Microsoft, Inc. Coherent is a trademark of Mark Williams Company. VMS is a trademark of Digital Equipment Corporation. C-ISAM and INFORMIX are trademarks of Relational Database Systems, Inc. Oracle is a trademark of Oracle Inc. PHACT is a trademark of Phact Associates. Logix is a trademark of Logical Software, Inc. IBM PC XT and AT are trademarks of International Business Machines Corporation. UNIFY is a trademark of Unify Corp.

cLINE™, Inc.

SAMPLE cENGLISH PROGRAM

```
IDENTIFICATIONS
  MODULE:   Mininame
  AUTHOR:   bcs
  DATE:     8/29/84
  REMARKS:  Sample cENGLISH program that adds first
            names to a file
END IDENTIFICATIONS

GLOBALS
  FIXED LENGTH 1 ans
  FIXED LENGTH 15 Fname
END GLOBALS

MAIN PROGRAM
  BEGIN
    CLEAR SCREEN
    SET ECHO OFF

    USE "NAMES"
    VIEW BY "ID_FNAME" ASCENDING

    AT 23,1 SAY "Add a record? Y or N"
    AT 23,25 ENTER ans USING "!"

    WHILE ans EQ "Y"
      CLEAR GETS
      AT 6,1 SAY "Enter first name"
      AT 6,20 GET Fname
      READ SCREEN

      INSERT
        Fname = Fname
      END INSERT

      AT 12,10 SAY "Welcome to cENGLISH," & Fname
      WAIT
      AT 14,10 SAY "HIT ANY KEY TO CONTINUE"
      STORE "                    " TO Fname
      STORE "   " TO ans
      AT 23,1 SAY "Add another record? Y or N"
      AT 23,30 ENTER ans USING "!"
      CLEAR ROW 1 THRU 23

    END WHILE

    AT 12,10 SAY "That's all for now!"
    UNUSE "NAMES"
    SET ECHO ON

  END PROGRAM
```

I'd like to know more about cENGLISH.
Please send further information.

Your Name ____________ Title ____________
Company ____________ Telephone ____________
Address ____________
City ____________ State ____________ Zip ____________
Check one: ☐ End User ☐ System House ☐ Dealer ☐ Distributor

Send to: cLINE Inc., 20 West Ontario, Chicago, IL 60610-3809
Telex 516315 Phone (312) 944-4510
In Canada: cLINE Canada, Inc. Complexe La Laurentienne,
425 St. Amable, Suite 165, Quebec, Canada G1R5E4
Phone (418) 524-4641

A4Q84

CIRCLE 26 ON READER SERVICE CARD

TRS-80 benchmark comparisons

	Sieve	Mysort
Source code size (bytes)	390	296
Assembly code size (bytes)	2,768	2,750
Executable code size (bytes)	3,129	3,139
Compile time (min:sec)	00:28	00:28
Assembly time (min:sec)[1]	02:53	02:59
Execution time (min:sec)[2]	01:52	00:32

1. EDAS is disk bound when compiling LC-generated code; these times reflect performance of a two-pass assembler taking all input from floppy disk files.
2. Times include total program load and run time. Actual sorting time on mysort was about 9 sec; remaining time includes array initialization and results reporting.

Table 5.

The **C Interpreter:**

Instant-C™

Programming in C has never been Faster.
Learning C will never be Easier.

Instant-C™ is an optimizing interpreter for the C language that can make programming in C three or more times faster than using old-fashioned compilers and loaders. The interpreter environment makes C as easy to use and learn as Basic. Yet ***Instant-C***™ is 20 to 50 times faster than interpreted Basic. This new interactive development environment gives you:

Instant Editing. The full-screen editor is built into ***Instant-C***™ for immediate use. You don't wait for a separate editor program to start up.

Instant Error Correction. You can check syntax in the editor. Each error message is displayed on the screen with the cursor set to the trouble spot, ready for your correction. Errors are reported clearly, by the editor, and only one at a time.

Instant Execution. *Instant-C*™ uses no assembler or loader. You can execute your program as soon as you finish editing.

Instant Testing. You can immediately execute any C statement or function, set variables, or evaluate expressions. Your results are displayed automatically.

Instant Symbolic Debugging. Watch execution by single statement stepping. Debugging features are built-in; you don't need to recompile or reload using special options.

Instant Loading. Directly generates .EXE or .CMD files at your request to create stand-alone versions of your programs.

Instant Floating Point. Uses 8087* co-processor if present.

Instant Compatibility. Follows K & R standards. Comprehensive standard library provided, with source code.

Instant Satisfaction. Get more done, faster, with better results. ***Instant-C***™ is available now, and works under PC-DOS, MS-DOS*, and CP/M-86*.

Find out how ***Instant-C***™ is changing the way that programming is done. ***Instant-C***™ is $500. Call or write for more information.

Rational
Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), 8087 (Intell Corp.), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

CIRCLE 56 ON READER SERVICE CARD

Manufacturer	Product
Alcor 1132 Commerce Richardson, Texas 75081 (214) 238-8554	Alcor C
BD Software P.O. Box 2368 Cambridge, Mass. 02238 (617) 576-3828	BDS C
C-Systems P.O. Box 3253 Fullerton, Calif. 92634 (714) 637-5362	C
C Ware P.O. Box C Sunnyvale, Calif. 94087 (408) 720-9696	DeSmet C
The Code Works Box 6905 Santa Barbara, Calif. 93160 (805) 683-1585	Q/C
Computer Innovations 980 Shrewsbury Ave. Tinton Falls, N.J. 07724 (201) 542-5920	Optimizing C86
Datalight 11557 8th Ave. N.E. Seattle, Wash. 98125 (206) 367-1803	Small-C
Digital Research 60 Garden Ct. Rd. P.O. Box DRI Monterey, Calif. 93942 (408) 646-6464	C
Ecosoft 6413 N. College Ave. Indianaplolis, Ind. 46220 (317) 255-6476	Eco-C
Jim E. Hendrix P.O. Box 8378 University, Miss. 38677 (601) 234-7508 eves.	Small-C
Introl 647 W. Virginia St. Milwaukee, Wis. 53204 (414) 276-2937	C
Lattice P.O. Box 3072 Glen Ellyn, Ill. 60138 (312) 858-7950	C
Manx Software Systems P.O. Box 51 Shrewsbury, N.J. 07701 (800) 221-0440	Aztec C
Mark Williams Co. 1430 W. Wrightwood Ave. Chicago, Ill. 60614 (800) 692-1700	MWC86
Microsoft P.O. Box 97200 Bellvue, Wash. 98009 (206) 828-8089	C
MiSosys Inc. P.O. Box 239 Sterling, Va. 22170-0239 (703) 450-4181	
Rational Systems P.O. Box 480 Natick, Mass. 01760 (617) 653-6194	Instant C
Software Toolworks 15233 Ventura Blvd. Ste. 1118 Sherman Oaks, Calif. 91403 (818) 986-4885	C/80
Supersoft P.O. Box 1628 Champaign, Ill. 61820 (217) 359-2112	C
Wizard Systems 11 Willow Ct. Arlington, Mass. 02174 (617) 641-2379	Wizard C

STRUCTURE FOR BASIC

With PROFESSIONAL PROGRAMMING ENVIRONMENT

ELIMINATES LINE NUMBERS

ALLOWS MULTI-LINE CONDITIONALS

PCDOS/MSDOS

BENDORF ASSOCIATES
6006 S. MAIN
P.O. BOX 5910
ROSWELL, NM
88201
505 347-5701
VISA/MASTERCARD

WORKS WITH BASICA INTERPRETER COMPILER

FULL ERROR LOGGING

PROGRAM LISTER

LABELED PROCEDURES

MACROS

SUB-ROUTINES

LIBRARIES

$49.95

CIRCLE 10 ON READER SERVICE CARD

C Users' Group

Over 40 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group
415 E. Euclid • Box 97
McPherson, KS 67460
(316) 241-1065

CIRCLE 43 ON READER SERVICE CARD

A general purpose programming language for string and list processing and all forms of non-numerical computation.

SNOBOL4+ —the entire SNOBOL4 language with its superb pattern-matching facilities • Strings over 32,000 bytes in length • Integer and floating point using 8087 or supplied emulator • ASCII, binary, sequential, and random-access I/O • Assembly Language interface • Compile new code during program execution • Create SAVE files • Program and data space up to 300K bytes RAM

With **ELIZA** & over 100 sample programs and functions

Have you tried SNOBOL4+?

MUSIC ANALYSIS • TEXT PROCESSING • SYMBOLIC MATHEMATICS • ARTIFICIAL INTELLIGENCE • COMPILER PROTOTYPING • CRYPTOGRAPHY • LINGUISTICS • LIST PROCESSING • GAMES •

For all 8086/88 PC/MS-DOS or CP/M-86 systems, 128K minimum 5¼" DSDD, specify DOS/CPM format

Send check, VISA, M/C to: **$95** plus $3 s/h

Catspaw, Inc.
P.O. Box 1123 • Salida, CO 81201 • 303/539-3884

CIRCLE 9 ON READER SERVICE CARD

* * * EASY TO USE * * *
Macro Programs for
Spellbinder™

NOW for PC-DOS

Customized Tables -- RAM disk with .BAT file. Toggle extended character set and printer control.

Our old standby, DearJohn, mailing list management with utilities. Still $67.50 CP/M or PC-DOS

"NEW" "All You Wanted to know about Spellbinder but Couldn't Find Out". A programmer's notebook of major commands and how to use. $10.00 per copy

We have a long list of custom macros for use in the office and by the programmer. Send $1.00 for catalog.

COMPUTER RESOURCES of WAIMEA
P.O. Box 1569 Kamuela, Hawaii 96743
(808) 885-7905

CIRCLE 14 ON READER SERVICE CARD

ADVERTISE

in the April issue of

COMPUTER LANGUAGE

Special distribution at the West Coast Computer Faire

Reservation deadline:
February 6th
Call (415) 957 9353

BDS C v1.5
For *FAST* Development

- Fastest compile to execute cycle of any CP/M 80® Compiler.
- Dynamic debugger.
- 180 page manual.

KS res. add 4% tax.
$120.00
Plus $2.50 Shipping & handling

TERMS: check, c.o.d., charge card.
(316) 431-0018

DMS Dedicated Micro Systems, Inc.
P.O. Box 481 Chanute, KS 66720

CP/M ® trademark of Digital Research

CIRCLE 21 ON READER SERVICE CARD

C Programmers
B-Trees
For
$75.00 + 2.00 Postage
Source Code Included

The Softfocus B-Trees record indexing library will help you develop sophisticated application programs. With Softfocus B-Trees you get:

- high speed file handling for up to 16.7 million records per file
- customizable BDS or K & R standard C source code
- no royalties on applications programs
- support random and sequential file access
- includes example programs

Softfocus
1277 Pallatine Dr., Oakville, Ont.
Canada L6H 1Z1 (416) 844-2610
CHARGEX VISA

CIRCLE 79 ON READER SERVICE CARD

CompuPro®
Users Group
Over 700 Members and Growing!

Join now to receive:
— Technical Newsletter
— Bulletin Board & RCPM
— Software Reviews
— New Product Announcements
— Hardware Help
— And More!

All for just $24/year!
Send check to:
C-PRO Users Group
P.O. Box 2146
Woodbridge, VA 22193

CIRCLE 61 ON READER SERVICE CARD

↓
→ *All* LOMAS DATA PRODUCTS' ←
16-Bit. S-100 Bus: Systems & Boards

* * * *FIREBALL DISCOUNTS* * * *
25% off List Price (UPS extra) for Cashier's Check or Money Order
+ + +
You save *MONEY* = We save *TIME*
+ + +
See → LOMAS ← ad in BYTE or request complete specifications

\ *LIGHTNING SPEED* /
80286-CPU & 80287-NDP: *$1458.75*

! ! *THUNDERING PERFORMANCE* ! !
80186 & I/O & 256K-DRAM & 5.25"/8" disk controller & CCP/M-86: *$1196.25*

Configured for *SanTec-S700* printer:
TeleVideo-GA970C VDT: *$1121.75*
Spellbinder-5.30 W/P: *$371.25*

HIERATIC COMPUTER SYSTEMS
BOX 133; MEDFORD, MA 02155
(617) 683-6540

CIRCLE 45 ON READER SERVICE CARD

ADVERTISER INDEX

	PAGE NO.	CIRCLE NO.
Alcor Systems	12	1
Amber Systems	4	2
BD Software	42	3
Bendorf & Associates	103	10
Blaise Computing Inc.	83	11
Borland International	6,7	6
Business Utility Software	86	17
C Journal	86	5
C Systems	59	16
C Users Group	103	43
C Ware	38	18
C-Line Inc.	101	26
C Pro Newsletter	103	61
Catspaw Inc.	103	9
Code Works (The)	24	42
CompuPro	COVER IV	12
CompuServe	2	7
Computer Innovations	41	8
Computer Resources of Waimea	103	14
Computing!	74	37
Compuview Products	81	40
Creative Solutions	96	28
DWB Associates	23	20
Data Base Decisions	47	30
Datalight	24	19
David Smith Software Co.	90	36
Dedicated Micro Systems Inc.	103	21
Ecosoft	90	22
Essential Software Inc.	32	27
FairCom	91	29
Greenleaf Software	88	44
HSC, Inc.	92	31
Harvard Softworks	85	47
Haventree Software	12	41
Hieratic Computer Systems	103	45
Introl Corp.	62	32
Korsmeyer Electronics Design	78	34
Laboratory Microsystems Inc.	91	35
Lahey Computer Systems Inc.	94	62
Lattice Inc.	64	53
Lifeboat Associates	52,53	64
Lomas Data Products	25	54
mbp Software & Systems Technology	10	39
MEF Environmental	89	15
MEF Environmental	86	55
Manx Software Systems	8	69
Megamax, Inc.	70	13
Mendocino Software Co. Inc.	70	48
Micro Methods	70	52
Micromotion	95	73
MicroTec Research Inc.	76	71
MicroWay	72	57
Mindbank Inc.	22	63
Next Generation Systems	66	74
Northwest Computer Algorythms	104	46
Oasys	63	75
Oasys	65	80
Opt-Tech Data Processing	86	66
Opt-Tech Data Processing	95	67
Parsec Research	95	49
Phoenix Computer Products Corp.	48	82
Plum Hall	67	50
Poor Person Software	58	51
Prentice Hall	16	83
ProCode	85	60
Programmer's Connection	97	84
Programmer's Shop (The)	60	98
QCAD	98	23
RR Software	COVER II	58
Rational Systems, Inc.	102	56
Realia, Inc.	93	76
Ream, Edward	98	-
SLR Systems	72	59
Seven Seas	98	4
Shaw American Technologies	58	92
Simpliway Products Co.	98	70
Soft Advances	85	77
Soft Craft Inc.	77	78
Softfocus	103	79
Software Horizons	84	25
Solution Systems	60	93
Solution Systems	60	99
Solution Systems	60	97
Spruce Technology Corp.	87	33
Steve Rank Inc.	68	87
Stride Micro	COVER III	72
Summit Software	68	88
Systems Guild	18	89
Thunder Software	88	65
UniPress	67	38
Western Wares	68	90
Whitesmiths Ltd.	1	85
Wizard Systems	70	86
Workman & Associates	78	68
Zeducomp	80	91

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

(LISP) FOR A.I.

UO-LISP Programming Environment
The Powerful Implementation of LISP
for MICRO COMPUTERS

LEARN LISP System (LLS.1)
(see description below *$39.95*
UO-LISP Programming Environment
Base Line System (BLS.1) *$49.95*

Includes: Interpreter, Compiler, Structure Editor, Extended Numbers, Trace, Pretty Print, various Utilities, and Manual with Usage Examples. **(BLS.1)** expands to support full system and products described below.

UO-LISP Programming Environment: The Usual LISP Interpreter Functions, Data Types and Extensions, Structure & Screen Editors, Compiler, Optimizer, LISP & Assembly Code Intermixing, Compiled Code Library Loader, I/O Support, Macros, Debug Tools, Sort & Merge, On-Line Help, Other Utility Packages, Hardware and Operating System Access, Session Freeze and Restart, Manual with Examples expands to over 350 pages. Other UO-LISP products include: LISPTEX text formatter, LITTLE META translator writing system, RLISP high level language, NLARGE algebra system. Prices vary with configurations beyond (BLS.1) please *send for FREE catalog.*

LEARN LISP System (LLS.1): Complete with LISP Tutorial Guide, Editor Tutorial Guide, System Manual with Examples, Full LISP Interpreter, On-Line Help and other Utilities. LEARN LISP fundamentals and programming techniques rapidly and effectively. This system does not permit expansion to include the compiler and other products listed above.

LISP Tutorial Support (LTS.1): Includes LISP and Structure Editor Tutorial Guides, On-line Help, and History Loop. This option adds a valuable learning tool to the UO-LISP Programming Environment (BLS.1). *Order (LTS.1) for* **$19.95.**

REQUIRES: UO-LISP Products run on most Z80 computers with CP/M, TRSDOS or TRSDOS compatible operating systems. The 8086 version available soon.

TO ORDER: Send Name, Address, Phone No., Computer Type, Disk Format Type, Package Price, 6.5% Tax (CA residents only), Ship & Handle fee of $3.00 inside U.S. & CN, $10 outside U.S., Check, Money Order, VISA and MasterCard accepted. With Credit Card include exp. date. Other configurations and products are ordered thru our *FREE catalog.*

Northwest Computer Algorithms
P.O. Box 90995, Long Beach, CA 90809 (213) 426-1893

CIRCLE 46 ON READER SERVICE CARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

SUBSCRIBE

COMPUTER
LANGUAGE

Subscribe to **COMPUTER LANGUAGE** today for only $24.95—over 30% savings off the single copy price.

☐ Yes, start my Subscription to COMPUTER LANGUAGE today. The cost is only $24.95 for 1 year (12 issues).

☐ I want to increase my savings even more—send me 2 years (24 issues) of COMPUTER LANGUAGE for only $39.95.

☐ Payment enclosed ☐ Bill me

Name ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Please allow 6–8 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canada orders $30.95 per year. Outside the U.S., $36.95/year for surface mail or $54.95/year for airmail.

BIF5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

SUBSCRIBE

COMPUTER
LANGUAGE

Subscribe to COMPUTER LANGUAGE today for only $24.95—over 30% savings off the single copy price.

☐ Yes, start my Subscription to COMPUTER LANGUAGE today. The cost is only $24.95 for 1 year (12 issues).

☐ I want to increase my savings even more—send me 2 years (24 issues) of COMPUTER LANGUAGE for only $39.95.

☐ Payment enclosed ☐ Bill me

Name ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Please allow 6–8 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canada orders $30.95 per year. Outside the U.S., $36.95/year for surface mail or $54.95/year for airmail.

BIF5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

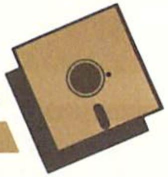
FIRST CLASS PERMIT NO. 27346 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

FREE INFORMATION



Name ______________________
Company ______________________
Address ______________________
City, State, Zip ______________________
Country ________ Telephone number ______________________

February issue. Not good if mailed after June 30, 1985.

Circle numbers for which you desire information.

1	11	21	31	41	51	61	71	81	91	101	111	121	131	141
2	12	22	32	42	52	62	72	82	92	102	112	122	132	142
3	13	23	33	43	53	63	73	83	93	103	113	123	133	143
4	14	24	34	44	54	64	74	84	94	104	114	124	134	144
5	15	25	35	45	55	65	75	85	95	105	115	125	135	145
6	16	26	36	46	56	66	76	86	96	106	116	126	136	146
7	17	27	37	47	57	67	77	87	97	107	117	127	137	147
8	18	28	38	48	58	68	78	88	98	108	118	128	138	148
9	19	29	39	49	59	69	79	89	99	109	119	129	139	149
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

I obtained this issue through:
- ☐ Subscription
- ☐ Computer Store
- ☐ Retail outlet
- ☐ Passed on by associate
- ☐ Other ________

Comments ______________________

Attn: Reader Service Dept.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

FREE INFORMATION

Name ______________________
Company ______________________
Address ______________________
City, State, Zip ______________________
Country ________ Telephone number ______________________

February issue. Not good if mailed after June 30, 1985.

Circle numbers for which you desire information.

1	11	21	31	41	51	61	71	81	91	101	111	121	131	141
2	12	22	32	42	52	62	72	82	92	102	112	122	132	142
3	13	23	33	43	53	63	73	83	93	103	113	123	133	143
4	14	24	34	44	54	64	74	84	94	104	114	124	134	144
5	15	25	35	45	55	65	75	85	95	105	115	125	135	145
6	16	26	36	46	56	66	76	86	96	106	116	126	136	146
7	17	27	37	47	57	67	77	87	97	107	117	127	137	147
8	18	28	38	48	58	68	78	88	98	108	118	128	138	148
9	19	29	39	49	59	69	79	89	99	109	119	129	139	149
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

I obtained this issue through:
- ☐ Subscription
- ☐ Computer Store
- ☐ Retail outlet
- ☐ Passed on by associate
- ☐ Other ________

Comments ______________________

Attn: Reader Service Dept.

"Despite the recent press notices, multiuser microcomputers aren't anything new!"

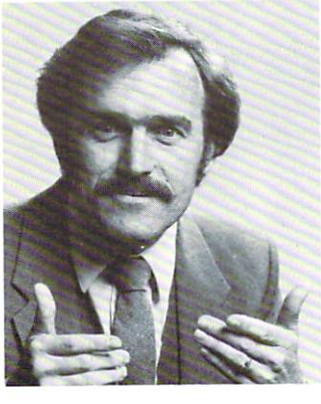
This is the first in a series of discussions with Rod Coleman, President of Stride Micro (formerly Sage Computer) on the 68000 multiuser market and its current environment.

Q: Why do you say that?

RC: "The technology to build a high performance multiuser system has been around for five years. And while some of the leaders in this industry have been pretending that micro multiuser didn't exist, we've been shipping complete systems for nearly three years. The benefits of multiuser are undeniable; it is more cost effective, and offers greater flexibility and utility. But until just recently, the marketing pressure to be compatible instead of being better, has blinded the industry."

Q: What do you mean?

RC: "Well, for example, the Motorola 68000 processor introduced 16/32-bit technology to the personal computer world a long time ago. It was fully capable of meeting high performance and multiuser design requirements in 1980. Instead of this trend taking off, most energy was spent promoting 8088/8086 products that were clearly inferior from a technical point of view. This phenomenon leads me to believe that they will soon rewrite the old proverb: 'Build a better mousetrap and the world will beat a path to your door,' but only if they can find the way through the marketing fog."



"A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it."

Q: Are things changing now?

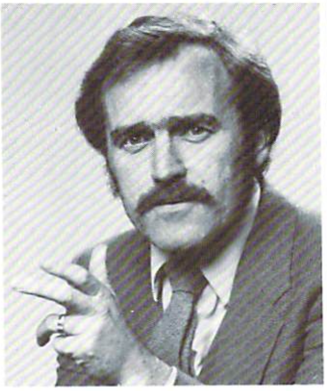
RC: "Yes and no. With the business world starting to take more and more interest in microcomputer solutions, the advantages of a solid multiuser system couldn't be kept hidden forever; companies like ours and a few others were beginning to make a dent. Instead of taking a fresh approach, some of the newest multiuser offerings will probably only give the technology an undeserved black eye! Multiuser is far more than the ability to plug in more terminals. It involves things like machine compatibility, fast processors, adequate memory, large storage capacities, backup features, networking, and operating system flexibility."

Q: Is this what makes the new Stride 400 Series different?

RC: "Exactly. That sounds self-serving, but it's true. Today a number of companies are introducing their first multiuser system. We've been building and shipping multiuser machines for almost three years. We know the pitfalls, we've fallen into some of them. But we have learned from our mistakes."

Q: Give me some examples.

RC: A hard disk is almost mandatory for any large multiuser installation. Yet, backing up a hard disk can be a nightmare if you only have floppies to work with. That's why we've added a tape backup option to all the larger Stride 400 Series machines. It's irresponsible for a manufacturer to market a multiuser system without such backup. Another good lesson was bus design. We started with one of our own designs, but learned that it's important not only to find a bus that is powerful, but also one that has good support and a strong future to serve tomorrow's needs. We think the VMEbus is the only design that meets both criteria and thus have made it a standard feature of every Stride 400 Series machine."



"The marketing pressure to be compatible instead of being better, has blinded the industry."

Q: What are some of the other unique features of the 400 Series?

RC: "A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it. Our systems are completely compatible with each other from the 420 model starting at $2900, through the 440, on to the powerful 460 which tops out near $60,000. Each system can talk to the others via the standard built-in local area network. Go ahead and compare this with others in the industry. You'll find their little machines don't talk to their big ones, or that the networking and multiuser are incompatible, or that they have different processors or operating systems, and so on."

Q: When you were still known as Sage Computer, you had a reputation for performance, is that still the case with the new Stride 400 Series?

RC: "Certainly, that's our calling card: 'Performance By Design.' Our new systems are actually faster; our standard processor is a 10 MHz 68000 running with no wait states. That gives us a 25% increase over the Sage models. And, we have a 12 MHz processor as an option. Let me add that speed isn't the only way to judge performance. I think it is also measured in our flexibility. We support a dozen different operating systems, not just one. And our systems service a wide variety of applications from the garage software developer to the corporate consumer running high volume business applications."

Q: Isn't that the same thing all manufacturers say in their ads?

RC: "Sure it is. But to use another over used-term, 'shop around'. We like to think of our systems as 'full service 68000 supermicrocomputers.' Take a look at everyone else's literature and then compare. When you examine cost, performance, flexibility, and utility, we don't think there's anyone else in the race. Maybe that's why we've shipped and installed more multiuser 68000 systems than anyone else."

STRIDE
MICRO

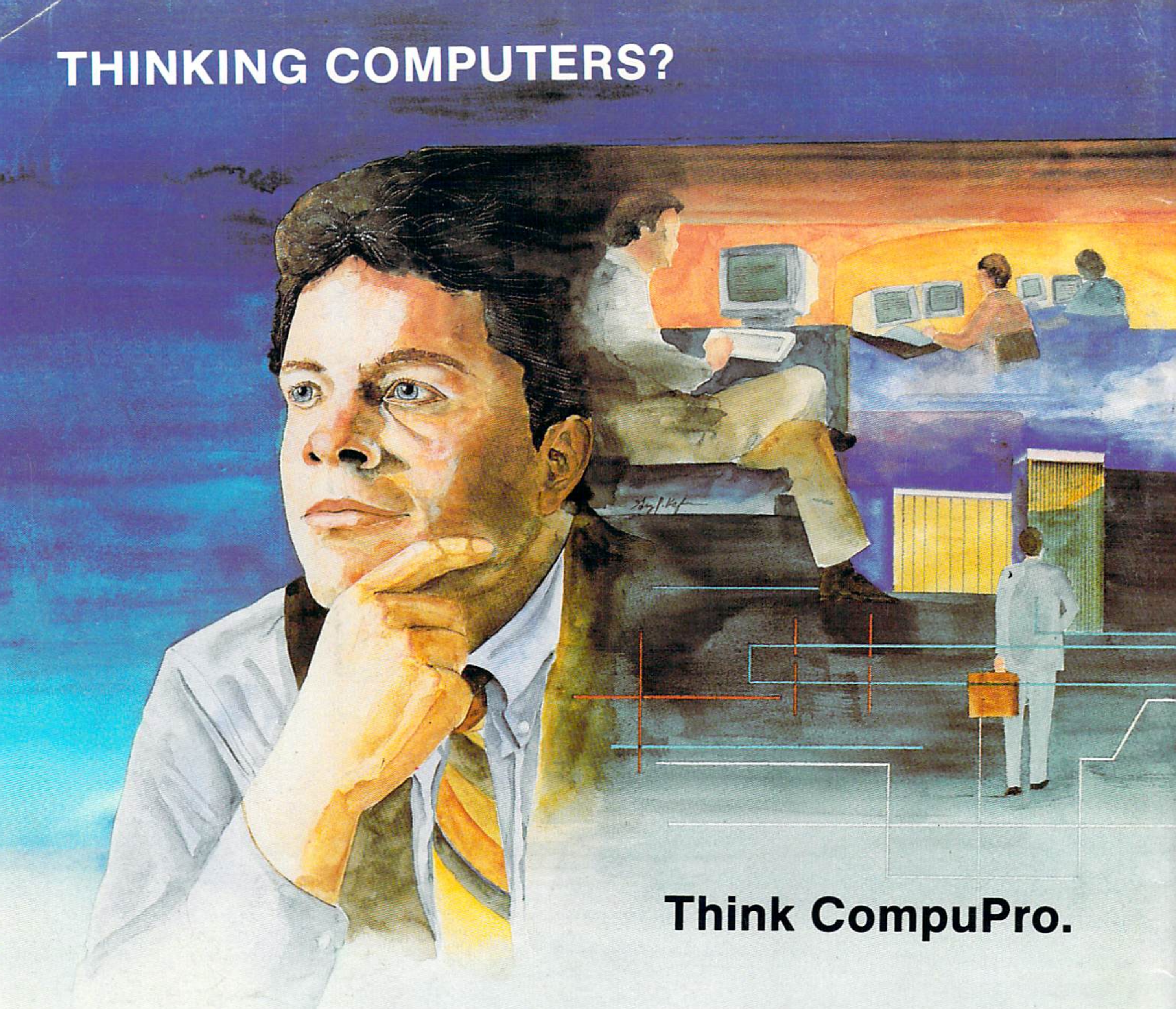
Formerly Sage Computer

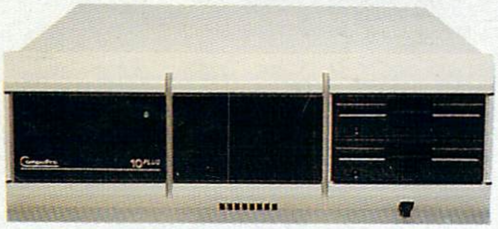
For more information on Stride or the location of the nearest Stride Dealer call or write us today. We'll also send you a free copy of our 32 page product catalog.

Corporate Offices:
4905 Energy Way
Reno, NV 89502
(702) 322-6868

Regional Offices:
Boston: (617) 229-6868
Dallas: (214) 392-7070

CIRCLE 72 ON READER SERVICE CARD


THINKING COMPUTERS?
Think CompuPro.

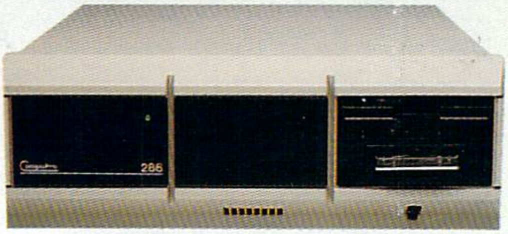


CompuPro 10*PLUS*

"... a coherent, relatively easy to use [multi-user] system with an impressive amount of power for the money and good support... alternative to stand alone—or even networked—computers, the CompuPro 10 has a lot going for it."

Personal Computing, 12/84

$7995.00 suggested list with 40 Mb hard disk



CompuPro 286

A faster version of our multi-user Model C, of which a recent review states "... hardware is high quality ... large software base... hundreds of serious business programs... versatility and adaptability to future hardware and software developments are unmatched..."

Business Computer Systems, 9/84

$9995.00 suggested list with 40Mb hard disk and cartridge tape

CIRCLE 12 ON READER SERVICE CARD

To arrange a demonstration, call **1-800-367-7816** (in California, **1-415-786-0909**).

©Viasyn 1985

Viasyn Corporation, 3506 Breakwater Court, Hayward, CA 94545